First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics

Meghyn Bienvenu
CNRS, Univ. Montpellier, Inria, France
meghyn@lirmm.fr

Peter Hansen and **Carsten Lutz**
University of Bremen, Germany
{hansen, clu}@informatik.uni-bremen.de

Frank Wolter
University of Liverpool, UK
frank@csc.liv.ac.uk

Abstract

We study FO-rewritability of conjunctive queries in the presence of ontologies formulated in a description logic between $\mathcal{EL}$ and Horn-$\mathcal{SHIF}$, along with related query containment problems. Apart from providing characterizations, we establish complexity results ranging from ExpTime via NExpTime to 2ExpTime, pointing out several interesting effects. In particular, FO-rewriting is more complex for conjunctive queries than for atomic queries when inverse roles are present, but not otherwise.

1 Introduction

When ontologies are used to enrich incomplete and heterogeneous data with a semantics and with background knowledge [Calvanese *et al.*, 2009; Kontchakov *et al.*, 2013; Bienvenu and Ortiz, 2015], efficient query anwering is a primary concern. Since classical database systems are unaware of ontologies and implementing new ontology-aware systems that can compete with these would be a huge effort, a main approach used today is *query rewriting*: the user query q and the ontology $\mathcal{O}$ are combined into a new query $q_{\mathcal{O}}$ that produces the same answers as q under $\mathcal{O}$ (over all inputs) and can be handed over to a database system for execution. Popular target languages for the query $q_{\mathcal{O}}$ include SQL and Datalog. In this paper, we concentrate on ontologies formulated in description logics (DLs) and on rewritability into SQL, which we equate with first-order logic (FO).

FO-rewritability in the context of query answering under DL ontologies was first studied in [Calvanese *et al.*, 2007]. Since FO-rewritings are not guaranteed to exist when ontologies are formulated in traditional DLs, the authors introduce the DL-Lite family of DLs specifically for the purpose of ontology-aware query answering using SQL database systems; in fact, the expressive power of DL-Lite is seriously restricted, in this way enabling existence guarantees for FO-rewritings. While DL-Lite is a successful family of DLs, there are many applications that require DLs with greater expressive power. The potential non-existence of FO-rewritings in this case is not necessarily a problem in practical applications. In fact, ontologies emerging from such applications typically use the available expressive means in a harmless way in the sense that efficient reasoning is often possible despite high worst-case complexity. One might thus hope that, in practice, FO-rewritings can often be constructed also beyond DL-Lite.

This hope was confirmed in [Bienvenu *et al.*, 2013; Hansen *et al.*, 2015], which consider the case where ontologies are formulated in a DL of the $\mathcal{EL}$ family [Baader *et al.*, 2005] and queries are atomic queries (AQs) of the form $A(x)$. To describe the obtained results in more detail, let an ontology-mediated query (OMQ) be a triple $(\mathcal{T}, \Sigma, q)$ with $\mathcal{T}$ a description logic TBox (representing an ontology), Σ an ABox signature (the set of concept and role names that can occur in the data), and q an actual query. Note that $\mathcal{T}$ and q might use symbols that do not occur in Σ; in this way, the TBox enriches the vocabulary available for formulating q. We use $(\mathcal{L}, \mathcal{Q})$ to denote the OMQ language that consists of all OMQs where $\mathcal{T}$ is formulated in the description logic $\mathcal{L}$ and q in the query language $\mathcal{Q}$. In [Bienvenu *et al.*, 2013], FO-rewritability is characterized in terms of the existence of certain tree-shaped ABoxes, covering a range of OMQ languages between $(\mathcal{EL}, \text{AQ})$ and (Horn-$\mathcal{SHI}$, AQ). On the one hand, this characterization is used to clarify the complexity of deciding whether a given OMQ is FO-rewritable, which turns out to be ExpTime-complete. On the other hand, it provides the foundations for developing practically efficient and complete algorithms for computing FO-rewritings. The latter was explored further in [Hansen *et al.*, 2015], where a novel type of algorithm for computing FO-rewritings of OMQs from $(\mathcal{EL}, \text{AQ})$ is introduced, crucially relying on the previous results from [Bienvenu *et al.*, 2013]. Its evaluation shows excellent performance and confirms the hope that, in practice, FO-rewritings almost always exist. In fact, rewriting fails in only 285 out of 10989 test cases.

A limitation of the discussed results is that they concern only AQs while in many applications, the more expressive conjunctive queries (CQs) are required. The aim of the current paper is thus to study FO-rewritability of OMQ languages based on CQs, considering ontology languages between $\mathcal{EL}$ and Horn-$\mathcal{SHIF}$. In particular, we provide characterizations of FO-rewritability in the required OMQ languages that are inspired by those in [Bienvenu *et al.*, 2013] (replacing tree-shaped ABoxes with a more general form of ABox), and we analyze the complexity of FO-rewritability using an automata-based approach. While practically efficient algorithms are out of the scope of this article, we believe that our work also lies important ground for the subsequent development of such

algorithms. Our approach actually *does* allow the construction of rewritings, but it is not tailored towards doing that in a practically efficient way. It turns out that the studied FO-rewritability problems are closely related to OMQ containment problems as considered in [Bienvenu *et al.*, 2012; Bourhis and Lutz, 2016]. In fact, being able to decide OMQ containment allows us to concentrate on connected CQs when deciding FO-rewritability, which simplifies technicalities considerably. For this reason, we also study characterizations and the complexity of query containment in the OMQ languages considered.

Our main complexity results are that FO-rewritability and containment are EXPTIME-complete for OMQ languages between $(\mathcal{EL}, \text{AQ})$ and $(\mathcal{ELHF}_\bot, \text{CQ})$ and 2EXPTIME-complete for OMQ languages between $(\mathcal{ELI}, \text{CQ})$ and (Horn-$\mathcal{SHIF}$, CQ). The lower bound for containment applies already when both OMQs share the same TBox. Replacing AQs with CQs thus results in an increase of complexity by one exponential in the presence of inverse roles (indicated by $\mathcal{I}$), but not otherwise. Note that the effect that inverse roles can increase the complexity of querying-related problems was known from expressive DLs of the $\mathcal{ALC}$ family [Lutz, 2008], but it has not previously been observed for Horn-DLs such as $\mathcal{ELI}$ and Horn-$\mathcal{SHIF}$. While 2EXPTIME might appear to be very high complexity, we are fortunately also able to show that the runtime is double exponential only in the size of the actual queries (which tends to be very small) while it is only single exponential in the size of the ontologies. We also show that the complexity drops to NEXPTIME when we restrict our attention to *rooted* CQs, that is, CQs which contain at least one answer variable and are connected. Practically relevant queries are typically of this kind.

A slight modification of our lower bounds yields new lower bounds for monadic Datalog containment. In fact, we close an open problem from [Chaudhuri and Vardi, 1994] by showing that containment of a monadic Datalog program in a rooted CQ is CONEXPTIME-complete. We also improve the 2EXPTIME lower bound for containment of a monadic Datalog program in a CQ from [Benedikt *et al.*, 2012] by showing that it already applies when the arity of EDB relations is bounded by two, rule bodies are tree-shaped, and there are no constants (which in this case correspond to nominals); the existing construction cannot achieve the latter two conditions simultaneously.

Full proofs are provided at http://www.informatik.uni-bremen.de/tdki/research/papers.html.

Related work. Pragmatic approaches to OMQ rewriting beyond DL-Lite often consider Datalog as a target language [Rosati, 2007; Pérez-Urbina *et al.*, 2010; Eiter *et al.*, 2012; Kaminski *et al.*, 2014; Trivela *et al.*, 2015]. These approaches might produce a non-recursive (thus FO) rewriting if it exists, but there are no guarantees. FO-rewritability of OMQs based on expressive DLs is considered in [Bienvenu *et al.*, 2014], and based on existential rules in [Baget *et al.*, 2011]. A problem related to ours is whether *all* queries are FO-rewritable when combined with a given TBox [Lutz and Wolter, 2012; Civili and Rosati, 2015]. There are several related works in the area of Datalog; recall that a Datalog program is bounded if and only if it is FO-rewritable [Ajtai and Gurevich, 1994]. Boundedness is known to be decidable [Cosmadakis *et al.*, 1988] and 2EXPTIME-complete [Benedikt *et al.*, 2015]; containment is also 2EXPTIME-complete [Cosmadakis *et al.*, 1988; Benedikt *et al.*, 2012]. OMQs from (Horn-$\mathcal{SHI}$, CQ) can be translated to monadic Datalog with an exponential blowup, functional roles (indicated by $\mathcal{F}$) are not expressible.

2 Preliminaries and Basic Observations

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be disjoint and countably infinite sets of *concept* and *role names*. A *role* is a role name r or an *inverse role* r^-, with r a role name. A *Horn-$\mathcal{SHIF}$ concept inclusion (CI)* is of the form $L \sqsubseteq R$, where L and R are concepts defined by the syntax rules

$$R, R' ::= \top \mid \bot \mid A \mid \neg A \mid R \sqcap R' \mid \neg L \sqcup R \mid \exists r.R \mid \forall r.R$$
$$L, L' ::= \top \mid \bot \mid A \mid L \sqcap L' \mid L \sqcup L' \mid \exists r.L$$

with A ranging over concept names and r over roles. In DLs, ontologies are formalized as TBoxes. A *Horn-$\mathcal{SHIF}$ TBox* $\mathcal{T}$ is a finite set of Horn-$\mathcal{SHIF}$ CIs, *functionality assertions* $\mathsf{func}(r)$, *transitivity assertions* $\mathsf{trans}(r)$, and *role inclusions (RIs)* $r \sqsubseteq s$, with r and s roles. It is standard to assume that functional roles are not transitive and neither are transitive roles included in them (directly or indirectly). We make the slighty stronger assumption that functional roles do not occur on the right-hand side of role inclusions at all. This assumption seems natural from a modeling perspective and mainly serves the purpose of simplifying constructions; all our results can be extended to the milder standard assumption. An *$\mathcal{ELIHF}_\bot$ TBox* is a *Horn-$\mathcal{SHIF}$ TBox* that contains neither transitivity assertions nor disjunctions in CIs, an *$\mathcal{ELI}$ TBox* is an $\mathcal{ELIHF}_\bot$ TBox that contains neither functionality assertions nor RIs, and an $\mathcal{ELHF}_\bot$ TBox is an $\mathcal{ELIHF}_\bot$ TBox that does not contain inverse roles.

An *ABox* is a finite set of *concept assertions* $A(a)$ and *role assertions* $r(a, b)$ where A is a concept name, r a role name, and a, b individual names from a countably infinite set $\mathsf{N_I}$. We sometimes write $r^-(a, b)$ instead of $r(b, a)$ and use $\mathsf{Ind}(\mathcal{A})$ to denote the set of all individual names used in $\mathcal{A}$. A *signature* is a set of concept and role names. We will often assume that the ABox is formulated in a prescribed signature, which we then call an *ABox signature*. An ABox that only uses concept and role names from a signature Σ is called a *Σ-ABox*.

The semantics of DLs is given in terms of *interpretations* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set (the *domain*) and $\cdot^\mathcal{I}$ is the *interpretation function*, assigning to each $A \in \mathsf{N_C}$ a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and to each $r \in \mathsf{N_R}$ a relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$ of a concept C in $\mathcal{I}$ is defined as usual, see [Baader *et al.*, 2003]. An interpretation $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$, a functionality assertion $\mathsf{func}(r)$ if $r^\mathcal{I}$ is a partial function, a transitivity assertion $\mathsf{trans}(r)$ if $r^\mathcal{I}$ is transitive, an RI $r \sqsubseteq s$ if $r^\mathcal{I} \subseteq s^\mathcal{I}$, a concept assertion $A(a)$ if $a \in A^\mathcal{I}$, and a role assertion $r(a, b)$ if $(a, b) \in r^\mathcal{I}$. We say that $\mathcal{I}$ is a *model of* a TBox or an ABox if it satisfies all inclusions and assertions in it. An ABox $\mathcal{A}$ is *consistent* with a TBox $\mathcal{T}$ if $\mathcal{A}$ and $\mathcal{T}$ have a common model. If α is a CI, RI, or functionality assertion, we write $\mathcal{T} \models \alpha$ if all models of $\mathcal{T}$ satisfy α.

A *conjunctive query (CQ)* takes the form $q = \exists \mathbf{x}\, \varphi(\mathbf{x}, \mathbf{y})$ with $\mathbf{x}, \mathbf{y}$ tuples of variables and φ a conjunction of atoms of

the form $A(x)$ and $r(x,y)$ that uses only variables from $\mathbf{x} \cup \mathbf{y}$. The variables in $\mathbf{y}$ are called *answer variables*, the *arity* of q is the length of $\mathbf{y}$, and q is *Boolean* if it has arity zero. An *atomic query (AQ)* is a conjunctive query of the form $A(x)$. A *union of conjunctive queries (UCQ)* is a disjunction of CQs that share the same answer variables. Ontology-mediated queries (OMQs) and the notation $(\mathcal{L}, \mathcal{Q})$ for OMQ languages were already defined in the introduction. We generally assume that if a role name r occurs in q and $\mathcal{T} \models s \sqsubseteq r$, then $\mathsf{trans}(s) \notin \mathcal{T}$. This is common since allowing transitive roles in the query poses serious additional complications, which are outside the scope of this paper; see e.g. [Bienvenu *et al.*, 2010; Gottlob *et al.*, 2013].

Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ, q of arity n, $\mathcal{A}$ a Σ-ABox and $\mathbf{a} \in \mathsf{Ind}(\mathcal{A})^n$. We write $\mathcal{A} \models Q(\mathbf{a})$ if $\mathcal{I} \models q(\mathbf{a})$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$. In this case, $\mathbf{a}$ is a *certain answer* to Q on $\mathcal{A}$. We use $\mathsf{cert}(Q, \mathcal{A})$ to denote the set of all certain answers to Q on $\mathcal{A}$.

A first-order query (FOQ) is a first-order formula φ constructed from atoms $A(x)$, $r(x,y)$, and $x = y$; here, concept names are viewed as unary predicates, role names as binary predicates, and predicates of other arity, function symbols, and constant symbols are not permitted. We write $\varphi(\mathbf{x})$ to indicate that the free variables of φ are among $\mathbf{x}$ and call $\mathbf{x}$ the *answer variables* of φ. The number of answer variables is the *arity* of φ and φ is *Boolean* if it has arity zero. We use $\mathsf{ans}(\mathcal{I}, \varphi)$ to denote the set of answers to the FOQ φ on the interpretation $\mathcal{I}$; that is, if φ is n-ary, then $\mathsf{ans}(\mathcal{I}, \varphi)$ contains all tuples $\mathbf{d} \in (\Delta^{\mathcal{I}})^n$ with $\mathcal{I} \models \varphi(\mathbf{d})$. To bridge the gap between certain answers and answers to FOQs, we sometime view an ABox $\mathcal{A}$ as an interpretation $\mathcal{I}_{\mathcal{A}}$, defined in the obvious way.

For any syntactic object O (such as a TBox, a query, an OMQ), we use $|O|$ to denote the *size* of O, that is, the number of symbols needed to write it (concept and role names counted as a single symbol).

Definition 1 (FO-rewriting)**.** An FOQ φ is an *FO-rewriting* of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ if $\mathsf{cert}(Q, \mathcal{A}) = \mathsf{ans}(\mathcal{I}_{\mathcal{A}}, \varphi)$ for all Σ-ABoxes $\mathcal{A}$ that are consistent with $\mathcal{T}$. If there is such a φ, then Q is *FO-rewritable*.

Example 2. *(1) Let $Q_0 = (\mathcal{T}_0, \Sigma_0, q_0(x,y))$, where $\mathcal{T}_0 = \{\exists r.A \sqsubseteq A, B \sqsubseteq \forall r.A\}$, $\Sigma_0 = \{r, A, B\}$ and $q_0(x,y) = B(x) \wedge r(x,y) \wedge A(y)$. Then $\varphi_0(x,y) = B(x) \wedge r(x,y)$ is an FO-rewriting of Q_0.*

We will see in Example 10 that the query Q_A obtained from Q_0 by replacing $q_0(x,y)$ with the AQ $A(x)$ is not FO-rewritable (due to the unbounded propagation of A via r-edges by $\mathcal{T}_0$). Thus, an FO-rewritable OMQ can give raise to AQ 'subqueries' that are not FO-rewritable.

(2) Let $Q_1 = (\mathcal{T}_1, \Sigma_1, q_1(x))$, where $\mathcal{T}_1 = \{\exists r.\exists r.A \sqsubseteq \exists r.A\}$, $\Sigma_1 = \{r, A\}$, and $q_1(x) = \exists y(r(x,y) \wedge A(y))$. Then Q_1 is not FO-rewritable (see again Example 10), but all AQ subqueries that Q_1 gives raise to are FO-rewritable.

The main reasoning problem studied in this paper is to decide whether a given OMQ $Q = (\mathcal{T}, \Sigma, q)$ is FO-rewritable. We assume without loss of generality that every symbol in Σ occurs in $\mathcal{T}$ or in q. We obtain different versions of this problem by varying the OMQ language used. Note that we have defined FO-rewritability relative to ABoxes that are consistent with the TBox. It is thus important for the user to know whether that is the case. Therefore, we also consider FO-rewritability of ABox inconsistency. More precisely, we say that *ABox inconsistency is FO-rewritable* relative to a TBox $\mathcal{T}$ and ABox signature Σ if there is a Boolean FOQ φ such that for every Σ-ABox $\mathcal{A}$, $\mathcal{A}$ is inconsistent with $\mathcal{T}$ iff $\mathcal{I}_{\mathcal{A}} \models \varphi()$.

Apart from FO-rewritability questions, we will also study OMQ containment. Let $Q_i = (\mathcal{T}_i, \Sigma, q_i)$ be two OMQs over the same ABox signature. We say that Q_1 *is contained in* Q_2, in symbols $Q_1 \subseteq Q_2$, if $\mathsf{cert}(Q_1, \mathcal{A}) \subseteq \mathsf{cert}(Q_2, \mathcal{A})$ holds for all Σ-ABoxes $\mathcal{A}$ that are consistent with $\mathcal{T}_1$ and $\mathcal{T}_2$.

We now make two basic observations that we use in an essential way in the remaining paper. We first observe that it suffices to concentrate on $\mathcal{ELIHF}_\bot$ TBoxes $\mathcal{T}$ in *normal form*, that is, all CIs are of one of the forms $A \sqsubseteq \bot, A \sqsubseteq \exists r.B, \top \sqsubseteq A, B_1 \sqcap B_2 \sqsubseteq A, \exists r.B \sqsubseteq A$ with A, B, B_1, B_2 concept names and r a role. We use $\mathsf{sig}(\mathcal{T})$ to denote the concept and role names that occur in $\mathcal{T}$.

Proposition 3. *Given a Horn-$\mathcal{SHIF}$ (resp. $\mathcal{ELHF}_\bot$) TBox $\mathcal{T}_1$ and ABox signature Σ, one can construct in polynomial time an $\mathcal{ELIHF}_\bot$ (resp. $\mathcal{ELHF}_\bot$) TBox $\mathcal{T}_2$ in normal form such that for every Σ-ABox $\mathcal{A}$,*

1. *$\mathcal{A}$ is consistent with $\mathcal{T}_1$ iff $\mathcal{A}$ is consistent with $\mathcal{T}_2$;*
2. *if $\mathcal{A}$ is consistent with $\mathcal{T}_1$, then for any CQ q that does not use symbols from $\mathsf{sig}(\mathcal{T}_2) \setminus \mathsf{sig}(\mathcal{T}_1)$, we have $\mathsf{cert}(Q_1, \mathcal{A}) = \mathsf{cert}(Q_2, \mathcal{A})$ where $Q_i = (\mathcal{T}_i, \Sigma, q)$.*

Theorem 3 yields polytime reductions of FO-rewritability in (Horn-$\mathcal{SHIF}, \mathcal{Q}$) to FO-rewritability in ($\mathcal{ELIHF}_\bot, \mathcal{Q}$) for any query language $\mathcal{Q}$, and likewise for OMQ containment and FO-rewritability of ABox inconsistency. It also tells us that, when working with $\mathcal{ELHF}_\bot$ TBoxes, we can assume normal form. Note that transitioning from (Horn-$\mathcal{SHF}, \mathcal{Q}$) to ($\mathcal{ELHF}_\bot, \mathcal{Q}$) is not as easy as in the case with inverse roles since universal restrictions on the right-hand side of concept inclusions cannot easily be eliminated; for this reason, we do not consider (Horn-$\mathcal{SHF}, \mathcal{Q}$). From now on, we work with TBoxes formulated in $\mathcal{ELIHF}_\bot$ or $\mathcal{ELHF}_\bot$ and assume without further notice that they are in normal form.

Our second observation is that, when deciding FO-rewritability, we can restrict our attention to connected queries provided that we have a way of deciding containment (for potentially disconnected queries). We use conCQ to denote the class of all connected CQs.

Theorem 4. *Let $\mathcal{L} \in \{\mathcal{ELIHF}_\bot, \mathcal{ELHF}_\bot\}$. Then FO-rewritability in $(\mathcal{L}, CQ)$ can be solved in polynomial time when there is access to oracles for containment in $(\mathcal{L}, \mathcal{Q})$ and for FO-rewritability in $(\mathcal{L}, conCQ)$.*

To prove Theorem 4, we observe that FO-rewritability of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ is equivalent to FO-rewritability of all OMQs $Q = (\mathcal{T}, \Sigma, q_c)$ with q_c a maximal connected component of q, excluding certain redundant such components (which can be identified using containment). Backed by Theorem 4, we generally assume connected queries when studying FO-rewritability, which allows to avoid unpleasant technical complications and is a main reason for studying FO-rewritability and containment in the same paper.

3 Main Results

In this section, we summarize the main results established in this paper. We start with the following theorem.

Theorem 5. *FO-rewritability and containment are*

1. 2EXPTIME*-complete for any OMQ language between* $(\mathcal{ELI}, CQ)$ *and* $(\textit{Horn-}\mathcal{SHIF}, CQ)$*, and*
2. EXPTIME*-complete for any OMQ language between* $(\mathcal{EL}, AQ)$ *and* $(\mathcal{ELHF}_\bot, CQ)$.

Moreover, given an OMQ from $(\textit{Horn-}\mathcal{SHIF}, CQ)$ *that is FO-rewritable, one can effectively construct a UCQ-rewriting.*

Like the subsequent results, Theorem 5 illustrates the strong relationship between FO-rewritability and containment. Note that inverse roles increase the complexity of both reasoning tasks. We stress that this increase takes place only when the actual queries are conjunctive queries, since FO-rewritability for OMQ languages with inverse roles and atomic queries is in EXPTIME [Bienvenu *et al.*, 2013].

The 2EXPTIME-completeness result stated in Point 1 of Theorem 5 might look discouraging. However, the situation is not quite as bad as it seems. To show this, we state the upper bound underlying Point 1 of Theorem 5 a bit more carefully.

Theorem 6. *Given OMQs* $Q_i = (\mathcal{T}_i, \Sigma_i, q_i)$*,* $i \in \{1, 2\}$*, from* $(\textit{Horn-}\mathcal{SHIF}, CQ)$*, it can be decided*

1. *in time* $2^{2^{p(|q_1| + \log(|\mathcal{T}_1|))}}$ *whether* Q_1 *is FO-rewritable and*
2. *in time* $2^{2^{p(|q_1| + |q_2| + \log(|\mathcal{T}_1| + |\mathcal{T}_2|))}}$ *whether* $Q_1 \subseteq Q_2$*,*

for some polynomial p*.*

Note that the runtime is double exponential only in the size of the actual queries q_1 and q_2, while it is only single exponential in the size of the TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$. This is good news since the size of q_1 and q_2 is typically very small compared to the sizes of $\mathcal{T}_1$ and $\mathcal{T}_2$. For this reason, it can even be reasonable to assume that the sizes of q_1 and q_2 are constant, in the same way in which the size of the query is assumed to be constant in data complexity. Note that, under this assumption, Theorem 6 yields EXPTIME upper bounds.

One other way to relativize the seemingly very high complexity stated in Point 1 of Theorem 5 is to observe that the lower bound proofs require the actual query to be Boolean or disconnected. In practical applications, though, typical queries are connected and have at least one answer variable. We call such CQs *rooted* and use rCQ to denote the class of all rooted CQs. Our last main result states that, when we restrict our attention to rooted CQs, then the complexity drops to CONEXPTIME.

Theorem 7. *FO-rewritability and containment are* CONEXPTIME*-complete in any OMQ language between* $(\mathcal{ELI}, rCQ)$ *and* $(\textit{Horn-}\mathcal{SHIF}, rCQ)$.

4 Semantic Characterization

The upper bounds stated in Theorems 5 and 6 are established in two steps. We first give characterizations of FO-rewritability in terms of the existence of certain (almost) tree-shaped ABoxes, and then utilize this characterization to design decision procedures based on alternating tree automata. The semantic characterizations are of independent interest.

An ABox $\mathcal{A}$ is *tree-shaped* if the undirected graph with nodes $\mathsf{Ind}(\mathcal{A})$ and edges $\{\{a, b\} \mid r(a, b) \in \mathcal{A}\}$ is acyclic and connected and $r(a, b) \in \mathcal{A}$ implies that (i) $s(a, b) \notin \mathcal{A}$ for all $s \neq r$ and (ii) $s(b, a) \notin \mathcal{A}$ for all role names s. For tree-shaped ABoxes $\mathcal{A}$, we often distinguish an individual used as the root, denoted with $\rho_\mathcal{A}$. $\mathcal{A}$ is *ditree-shaped* if the directed graph with nodes $\mathsf{Ind}(\mathcal{A})$ and edges $\{(a, b) \mid r(a, b) \in \mathcal{A}\}$ is a tree and $r(a, b) \in \mathcal{A}$ implies (i) and (ii). The (unique) root of a ditree-shaped ABox $\mathcal{A}$ is also denoted with $\rho_\mathcal{A}$.

An ABox $\mathcal{A}$ is a *pseudo tree* if it is the union of ABoxes $\mathcal{A}_0, \dots, \mathcal{A}_k$ that satisfy the following conditions:

1. $\mathcal{A}_1, \dots, \mathcal{A}_k$ are tree-shaped;
2. $k \leq |\mathsf{Ind}(\mathcal{A}_0)|$;
3. $\mathcal{A}_i \cap \mathcal{A}_0 = \{\rho_{\mathcal{A}_i}\}$ and $\mathsf{Ind}(\mathcal{A}_i) \cap \mathsf{Ind}(\mathcal{A}_j) = \emptyset$, for $1 \leq i < j \leq k$.

We call $\mathcal{A}_0$ the *core* of $\mathcal{A}$ and $\mathcal{A}_1, \dots, \mathcal{A}_k$ the *trees* of $\mathcal{A}$. The *width* of $\mathcal{A}$ is $|\mathsf{Ind}(\mathcal{A}_0)|$, its *depth* is the depth of the deepest tree of $\mathcal{A}$, and its *outdegree* is the maximum outdegree of the ABoxes $\mathcal{A}_1, \dots, \mathcal{A}_k$. For a pseudo tree ABox $\mathcal{A}$ and $\ell \geq 0$, we write $\mathcal{A}|_{\leq \ell}$ to denote the restriction of $\mathcal{A}$ to the individuals whose minimal distance from a core individual is at most ℓ, and analogously for $\mathcal{A}|_{>\ell}$. A *pseudo ditree ABox* is defined analogously to a pseudo tree ABox, except that $\mathcal{A}_1, \dots, \mathcal{A}_k$ must be ditree-shaped.

When studying FO-rewritability and containment, we can restrict our attention to pseudo tree ABoxes, and even to pseudo ditree ABoxes when the TBox does not contain inverse roles. The following statement makes this precise for the case of containment. Its proof uses unraveling and compactness.

Proposition 8. *Let* $Q_i = (\mathcal{T}_i, \Sigma, q_i)$*,* $i \in \{1, 2\}$*, be OMQs from* $(\mathcal{ELIHF}_\bot, CQ)$*. Then* $Q_1 \not\subseteq Q_2$ *iff there is a pseudo tree* Σ*-ABox* $\mathcal{A}$ *of outdegree at most* $|\mathcal{T}_1|$ *and width at most* $|q_1|$ *that is consistent with both* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *and a tuple* $\mathbf{a}$ *from the core of* $\mathcal{A}$ *such that* $\mathcal{A} \models Q_1(\mathbf{a})$ *and* $\mathcal{A} \not\models Q_2(\mathbf{a})$.

If Q_1, Q_2 *are from* $(\mathcal{ELHF}_\bot, CQ)$*, then we can find a pseudo ditree ABox with these properties.*

We now establish a first version of the announced characterizations of FO-rewritability. Like Proposition 8, they are based on pseudo tree ABoxes.

Theorem 9. *Let* $Q = (\mathcal{T}, \Sigma, q)$ *be an OMQ from* $(\mathcal{ELIHF}_\bot, conCQ)$*. If the arity of* q *is at least one, then the following conditions are equivalent:*

1. Q *is FO-rewritable;*
2. *there is a* $k \geq 0$ *such that for all pseudo tree* Σ*-ABoxes* $\mathcal{A}$ *that are consistent with* $\mathcal{T}$ *and of outdegree at most* $|\mathcal{T}|$ *and width at most* $|q|$*: if* $\mathcal{A} \models Q(\mathbf{a})$ *with* $\mathbf{a}$ *from the core of* $\mathcal{A}$*, then* $\mathcal{A}|_{\leq k} \models Q(\mathbf{a})$*;*

If q *is Boolean, this equivalence holds with (2.) replaced by*

2′. *there is a* $k \geq 0$ *such that for all pseudo tree* Σ*-ABoxes* $\mathcal{A}$ *that are consistent with* $\mathcal{T}$ *and of outdegree at most* $|\mathcal{T}|$ *and of width at most* $|q|$*: if* $\mathcal{A} \models Q$*, then* $\mathcal{A}|_{>0} \models Q$ *or* $\mathcal{A}|_{\leq k} \models Q$.

If Q *is from* $(\mathcal{ELHF}_\bot, conCQ)$*, then the above equivalences hold also when pseudo tree* Σ*-ABoxes are replaced with pseudo ditree* Σ*-ABoxes.*

The proof of Proposition 8 gives a good intuition of why FO-rewritability can be characterized in terms of ABoxes that are pseudo trees. In fact, the proof of "2 $\Rightarrow$ 1" of Theorem 9 is similar to the proof of Proposition 8. The proof of "1 $\Rightarrow$ 2" uses locality arguments in the form of Ehrenfeucht-Fraïssé games. The following examples further illustrate Theorem 9.

Example 10. *(1) Non FO-rewritability of the OMQs Q_A and Q_1 from Example 2 is shown by refuting Condition 2 in Theorem 9: let $\mathcal{A}_k = \{r(a_0, a_1), \dots, r(a_k, a_{k+1}), A(a_{k+1})\}$, for all $k \geq 0$. Then $\mathcal{A}_k \models Q(a_0)$ but $\mathcal{A}_k|_{\leq k} \not\models Q(a_0)$ for $Q \in \{Q_A, Q_1\}$.*

(2) Theorem 9 only holds for connected CQs: consider $Q_2 = (\mathcal{T}_2, \Sigma_2, q_2)$, where $\mathcal{T}_2$ is the empty TBox, $\Sigma_2 = \{A, B\}$, and $q_2 = \exists x \exists y (A(x) \wedge B(y))$. Q_2 is FO-rewritable (q_2 itself is a rewriting), but Condition $2'$ does not hold: for $\mathcal{B}_k = \{A(a_0), R(a_0, a_1, \dots, R(a_k, a_{k+1}), B(a_{k+1})\}$ we have $\mathcal{B}_k \models Q_2$ but $\mathcal{B}_k|_{>0} \not\models Q_2$ and $\mathcal{B}_k|_{\leq k} \not\models Q_2$.

(3) The modification $2'$ of Condition 2 is needed to characterize FO-rewritability of Boolean OMQs: obtain Q_B from Q_2 by replacing q_2 with $\exists x B(x)$. Then Q_B is FO-rewritable, but the ABoxes $\mathcal{B}_k$ show that Condition 2 does not hold.

Theorem 9 does not immediately suggest a decision procedure for FO-rewritability since there is no bound on the depth of the pseudo tree ABoxes $\mathcal{A}$ used. The next result establishes such a bound.

Theorem 11. *Let $\mathcal{T}$ be an $\mathcal{ELIHF}_\bot$ TBox. Then Theorem 9 still holds with the following modifications:*

1. *if q is not Boolean or $\mathcal{T}$ is an $\mathcal{ELHF}_\bot$ TBox, "there is a $k \geq 0$" is replaced with "for $k = |q| + 2^{4(|\mathcal{T}|+|q|)^2}$";*
2. *if q is Boolean, "there is a $k \geq 0$" is replaced with "for $k = |q| + 2^{4(|\mathcal{T}|+2^{|q|})^2}$".*

The proof of Theorem 11 uses a pumping argument based on derivations of concept names in the pumped ABox by $\mathcal{T}$. Due to the presence of inverse roles, this is not entirely trivial and uses what we call *transfer sequences*, describing the derivation history at a point of an ABox. Together with the proof of Theorem 9, Theorem 11 gives rise to an algorithm that constructs actual rewritings when they exist.

5 Constructing Automata

We show that Proposition 8 and Theorem 11 give rise to automata-based decision procedures for containment and FO-rewritability that establish the upper bounds stated in Theorems 5 and 6. By Theorem 4, it suffices to consider connected queries in the case of FO-rewritability. We now observe that we can further restrict our attention to Boolean queries. We use BCQ (resp. conBCQ) to denote the class of all Boolean CQs (resp. connected Boolean CQs).

Lemma 12. *Let $\mathcal{L} \in \{\mathcal{ELIHF}_\bot, \mathcal{ELHF}_\bot\}$. Then*

1. *FO-rewritability in $(\mathcal{L}, \text{conCQ})$ can be reduced in polytime to FO-rewritability in $(\mathcal{L}, \text{conBCQ})$;*
2. *Containment in $(\mathcal{L}, \text{CQ})$ can be reduced in polytime to containment in $(\mathcal{L}, \text{BCQ})$.*

The decision procedures rely on building automata that accept pseudo tree ABoxes which witness non-containment and non-FO-rewritability as stipulated by Proposition 8 and Theorem 11, respectively. We first have to encode pseudo tree ABoxes in a suitable way.

A *tree* is a non-empty (and potentially infinite) set $T \subseteq \mathbb{N}^*$ closed under prefixes. We say that T is *m-ary* if for every $x \in T$, the set $\{i \mid x \cdot i \in T\}$ is of cardinality at most m. For an alphabet Γ, a *Γ-labeled tree* is a pair (T, L) with T a tree and $L : T \to \Gamma$ a node labeling function. Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from $(\mathcal{ELIHF}_\bot, \text{conBCQ})$. We encode pseudo tree ABoxes of width at most $|q|$ and outdegree at most $|\mathcal{T}|$ by $(|\mathcal{T}| \cdot |q|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled trees, where Σ_ε is an alphabet used for labeling root nodes and Σ_N is for non-root nodes.

The alphabet Σ_ε consists of all Σ-ABoxes $\mathcal{A}$ such that $\mathsf{Ind}(\mathcal{A})$ only contains individual names from a fixed set $\mathsf{Ind}_{\mathsf{core}}$ of size $|q|$ and $\mathcal{A}$ satisfies all functionality statements in $\mathcal{T}$. The alphabet Σ_N consists of all subsets $\Theta \subseteq (\mathsf{N_C} \cap \Sigma) \uplus \{r, r^- \mid r \in \mathsf{N_R} \cap \Sigma\} \uplus \mathsf{Ind}_{\mathsf{core}}$ that contain exactly one (potentially inverse) role and at most one element of $\mathsf{Ind}_{\mathsf{core}}$. A $(|\mathcal{T}| \cdot |q|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree is *proper* if (i) the root node is labeled with a symbol from Σ_ε, (ii) each child of the root is labeled with a symbol from Σ_N that contains an element of $\mathsf{Ind}_{\mathsf{core}}$, (iii) every other non-root node is labeled with a symbol from Σ_N that contains no individual name, and (iv) every non-root node has at most $|q|$ successors and (v) for every $a \in \mathsf{Ind}_{\mathsf{core}}$, the root node has at most $|q|$ successors whose label includes a.

A proper $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree (T, L) represents a pseudo tree ABox $\mathcal{A}_{(T,L)}$ whose individuals are those in the ABox $\mathcal{A}$ that labels the root of T plus all non-root nodes of T, and whose assertions are

$$\begin{aligned}
&\mathcal{A} \cup \{A(x) \mid A \in L(x)\} \\
&\cup \{r(b, x) \mid \{b, r\} \subseteq L(x)\} \cup \{r(x, b) \mid \{b, r^-\} \subseteq L(x)\} \\
&\cup \{r(x, y) \mid r \in L(y), y \text{ is a child of } x, L(x) \in \Sigma_N\} \\
&\cup \{r(y, x) \mid r^- \in L(y), y \text{ is a child of } x, L(x) \in \Sigma_N\}.
\end{aligned}$$

As the automaton model, we use two-way alternating parity automata on finite trees (TWAPAs). As usual, $L(\mathfrak{A})$ denotes the tree language accepted by the TWAPA $\mathfrak{A}$. Our central observation is the following.

Proposition 13. *For every OMQ $Q = (\mathcal{T}, \Sigma, q)$ from $(\mathcal{ELIHF}_\bot, \text{BCQ})$, there is a TWAPA*

1. *$\mathfrak{A}_Q$ that accepts a $(|\mathcal{T}| \cdot |q|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree (T, L) iff it is proper, $\mathcal{A}_{(T,L)}$ is consistent with $\mathcal{T}$, and $\mathcal{A}_{(T,L)} \models Q$;*

   $\mathfrak{A}_Q$ has at most $2^{p(|q|+\log(|\mathcal{T}|))}$ states, and at most $p(|q| + |\mathcal{T}|)$ states if $\mathcal{T}$ is an $\mathcal{ELHF}_\bot$ TBox, p a polynomial.
2. *$\mathfrak{A}_\mathcal{T}$ that accepts a $(|\mathcal{T}| \cdot |q|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree (T, L) iff it is proper and $\mathcal{A}_{(T,L)}$ is consistent with $\mathcal{T}$.*

   $\mathfrak{A}_\mathcal{T}$ has at most $p(|\mathcal{T}|)$ states, p a polynomial.

We can construct $\mathfrak{A}_Q$ and $\mathfrak{A}_\mathcal{T}$ in time polynomial in their size.

The construction of the automata in Proposition 13 uses forest decompositions of the CQ q as known for example from [Lutz, 2008]. The difference in automata size between $\mathcal{ELIHF}_\bot$ and $\mathcal{ELHF}_\bot$ is due to the different number of tree-shaped subqueries that can arise in these decompositions.

To decide $Q_1 \subseteq Q_2$ for OMQs $Q_i = (\mathcal{T}_i, \Sigma, q_i)$, $i \in \{1, 2\}$, from $(\mathcal{ELIHF}_\bot, \text{BCQ})$, by Proposition 8 it suffices to decide whether $L(\mathfrak{A}_{Q_1}) \cap L(\mathfrak{A}_{\mathcal{T}_2}) \subseteq L(\mathfrak{A}_{Q_2})$. Since this question can be polynomially reduced to a TWAPA emptiness check and the latter can be executed in time single exponential in the number of states, this yields the upper bounds for containment stated in Theorems 5 and 6.

To decide non-FO-rewritability of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ from $(\mathcal{ELIHF}_\bot, \text{conBCQ})$, by Theorem 11 we need to decide whether there is a pseudo tree Σ-ABox $\mathcal{A}$ of outdegree at most $|\mathcal{T}|$ and width at most $|q|$ that is consistent with $\mathcal{T}$ and satisfies (i) $\mathcal{A} \models Q$, (ii) $\mathcal{A}|_{>0} \not\models Q$, and (iii) $\mathcal{A}|_{\leq k} \not\models Q$ where $k = |q| + 2^{4(|\mathcal{T}|+2^{|q|})^2}$. For consistency with $\mathcal{T}$ and for (i), we use the automaton $\mathfrak{A}_Q$ from Proposition 13. To achieve (ii) and (iii), we amend the tree alphabet $\Sigma_\varepsilon \cup \Sigma_n$ with additional labels that implement a counter which counts up to k and annotate each node in the tree with its depth (up to k). We then complement $\mathfrak{A}_Q$ (which for TWAPAs can be done in polynomial time), relativize the resulting automaton to all but the first level of the input ABox for (ii) and to the first k levels for (iii), and finally intersect all automata and check emptiness. This yields the upper bounds for FO-rewritability stated in Theorems 5 and 6.

As remarked in the introduction, apart from FO-rewritability of an OMQ $(\mathcal{T}, \Sigma, q)$ we should also be interested in FO-rewritability of ABox inconsistency relative to $\mathcal{T}$ and Σ. We close this section with noting that an upper bound for this problem can be obtained from Point 2 of Proposition 13 since TWAPAs can be complemented in polynomial time. A matching lower bound can be found in [Bienvenu *et al.*, 2013].

Theorem 14. *In $\mathcal{ELIHF}_\bot$, FO-rewritability of ABox inconsistency is* ExpTime*-complete.*

6 Rooted Queries and Lower Bounds

We first consider the case of rooted queries and establish the upper bound in Theorem 7.

Theorem 15. *FO-rewritability and containment in $(\mathcal{ELIHF}_\bot, rCQ)$ are in* coNExpTime*.*

Because of space limitations, we confine ourselves to a brief sketch, concentrating on FO-rewritability. By Point 1 of Theorem 11, deciding non-FO-rewritability of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ from $(\mathcal{ELIHF}_\bot, \text{rCQ})$ comes down to checking the existence of a pseudo tree Σ-ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$ and such that $\mathcal{A} \models Q(\mathbf{a})$ and $\mathcal{A}|_{\leq k} \not\models Q(\mathbf{a})$ for some tuple of individuals $\mathbf{a}$ from the core of $\mathcal{A}$, for some suitable k. Recall that $\mathcal{A} \models Q(\mathbf{a})$ if and only if there is a homomorphism h from q to the pseudo tree-shaped canonical model of $\mathcal{T}$ and $\mathcal{A}$ that takes the answer variables to $\mathbf{a}$. Because $\mathbf{a}$ is from the core of $\mathcal{A}$ and q is rooted, h can map existential variables in q only to individuals from $\mathcal{A}|_{|q|}$ and to the anonymous elements in the subtrees below them. To decide the existence of $\mathcal{A}$, we can thus guess $\mathcal{A}|_{|q|}$ together with sets of concept assertions about individuals in $\mathcal{A}|_{|q|}$ that can be inferred from $\mathcal{A}$ and $\mathcal{T}$, and from $\mathcal{A}|_{\leq k}$ and $\mathcal{T}$. We can then check whether there is a homomorphism h as described, without access to the full ABoxes $\mathcal{A}$ and $\mathcal{A}|_{\leq k}$. It remains to ensure that the guessed initial part $\mathcal{A}_{|q|}$ can be extended to $\mathcal{A}$ such that the entailed concept assertions are precisely those that were guessed, by attaching tree-shaped ABoxes to individuals on level $|q|$. This can be done by a mix of guessing and automata techniques.

We next establish the lower bounds stated in Theorems 5 and 7. For Theorem 5, we only prove a lower bound for Point 1 as the one in Point 2 follows from [Bienvenu *et al.*, 2013].

Theorem 16. *Containment and FO-rewritability are*

1. coNExpTime*-hard in $(\mathcal{ELI}, rCQ)$ and*
2. 2ExpTime*-hard in $(\mathcal{ELI}, CQ)$.*

The results for containment apply already when both OMQs share the same TBox.

Point 1 is proved by reduction of the problem of tiling a torus of exponential size, and Point 2 is proved by reduction of the word problem of exponentially space-bounded alternating Turing machines (ATMs). The proofs use queries similar to those introduced in [Lutz, 2008] to establish lower bounds on the complexity of query answering in the expressive OMQ languages $(\mathcal{ALCI}, \text{rCQ})$ and $(\mathcal{ALCI}, \text{CQ})$. A major difference to the proofs in [Lutz, 2008] is that we represent torus tilings / ATM computations in the ABox that witnesses non-containment or non-FO-rewritability, instead of in the 'anonymous part' of the model created by existential quantifiers.

The proof of Point 2 of Theorem 16 can be modified to yield new lower bounds for monadic Datalog containment. Recall that the rule body of a Datalog program is a CQ. *Tree-shapedness* of a CQ q is defined in the same way as for an ABox in Section 4, that is, q viewed as an undirected graph must be a tree without multi-edges.

Theorem 17. *For monadic Datalog programs which contain no EDB relations of arity larger than two and no constants, containment*

1. *in a rooted CQ is* coNExpTime*-hard;*
2. *in a CQ is* 2ExpTime*-hard, even when all rule bodies are tree-shaped.*

Point 1 closes an open problem from [Chaudhuri and Vardi, 1994], where a coNExpTime upper bound for containment of a monadic Datalog program in a rooted UCQ was proved and the lower bound was left open. Point 2 further improves a lower bound from [Benedikt *et al.*, 2012] which also does not rely on EDB relations of arity larger than two, but requires that rule bodies are not tree-shaped or constants are present (which, in this case, correspond to nominals in the DL world).

7 Conclusion

A natural next step for future work is to use the techniques developed here for devising practically efficient algorithms that construct actual rewritings, which was very successful in the AQ case [Hansen *et al.*, 2015].

An interesting open theoretical question is the complexity of FO-rewritability and containment for the OMQ languages considered in this paper in the special case when the ABox signature contains all concept and role names.

Acknowledgements. Bienvenu was supported by ANR project PAGODA (12-JS02-007-01), Hansen and Lutz by ERC grant 647289, Wolter by EPSRC UK grant EP/M012646/1.

References

[Ajtai and Gurevich, 1994] Miklós Ajtai and Yuri Gurevich. Datalog vs First-Order Logic. J. Comput. Syst. Sci.,49(3): 562–588, 1994.

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, pages 364–369, 2005.

[Baget *et al.*, 2011] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

[Benedikt *et al.*, 2012] Michael Benedikt, Pierre Bourhis, and Pierre Senellart. Monadic Datalog Containment. In *Proc. of ICALP*, pages 79–91, 2012.

[Benedikt *et al.*, 2015] Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The Complexity of Boundedness for Guarded Logics. In *Proc. of LICS*, pages 293–304, 2015.

[Bienvenu and Ortiz, 2015] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Proc. of Reasoning Web*, volume 9203 of *LNCS*, pages 218–307, 2015.

[Bienvenu *et al.*, 2010] Meghyn Bienvenu, Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Simkus. Query answering in the description logic S. In *Proc. of DL*, volume 573 of *CEUR-WS,* 2010.

[Bienvenu *et al.*, 2012] Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. Query containment in description logics reconsidered. In *Proc of KR*, pages 221–231, 2012.

[Bienvenu *et al.*, 2013] Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. First order-rewritability of atomic queries in Horn description logics. In *Proc. of IJCAI*, pages 754–760, 2013.

[Bienvenu *et al.*, 2014] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: a study through disjunctive datalog, CSP, and MMSNP. *Proc. of TODS*, 39, 2014.

[Bourhis and Lutz, 2016] Pierre Bourhis and Carsten Lutz. Containment in monadic disjunctive datalog, MMSNP, and expressive description logics. In *Proc. of KR*, 2016.

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[Calvanese *et al.*, 2009] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The DL-Lite approach. In *Proc. of Reasoning Web*, volume 5689 of *LNCS*, pages 255–356, 2009.

[Chaudhuri and Vardi, 1994] Surajit Chaudhuri and Moshe Y. Vardi. On the complexity of equivalence between recursive and nonrecursive datalog programs In *Proc. of PODS*, pages 107–116, 1994.

[Civili and Rosati, 2015] Cristina Civili and Riccardo Rosati. On the first-order rewritability of conjunctive queries over binary guarded existential rules. In *Proc. of CILC*, volume 1459 of *CEUR-WS*, pages 25–30, 2015.

[Cosmadakis *et al.*, 1988] Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *Proc. of STOC*, pages 477–490, 1988.

[Eiter *et al.*, 2012] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI*, 2012.

[Gottlob *et al.*, 2013] Georg Gottlob, Andreas Pieris, and Lidia Tendera. Querying the guarded fragment with transitivity. In *Proc. of ICALP*, volume 7966 of *LNCS*, pages 287–298, 2013.

[Hansen *et al.*, 2015] Peter Hansen, Carsten Lutz, Inanç Seylan, and Frank Wolter. Efficient query rewriting in the description logic EL and beyond. In *Proc. of IJCAI*, pages 3034–3040, 2015.

[Kaminski *et al.*, 2014] Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. Computing datalog rewritings for disjunctive datalog programs and description logic ontologies. In *Proc. of RR*, pages 76–91, 2014.

[Kontchakov *et al.*, 2013] Roman Kontchakov, Mariano Rodriguez-Muro, and Michael Zakharyaschev. Ontology-based data access with databases: A short course. In *Proc. of Reasoning Web*, pages 194–229, 2013.

[Lutz and Wolter, 2012] Carsten Lutz and Frank Wolter. Non-uniform data complexity of query answering in description logics. In *Proc. of KR*, 2012.

[Lutz, 2008] Carsten Lutz. The complexity of conjunctive query answering in expressive description logics. In *Proc. of IJCAR*, volume 5195 of *LNCS*, pages 179–193, 2008.

[Pérez-Urbina *et al.*, 2010] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. Tractable query answering and rewriting under description logic constraints. *J. Applied Logic*, 8(2):186–209, 2010.

[Rosati, 2007] Riccardo Rosati. On conjunctive query answering in EL. In *Proc. of DL*, pages 451–458, 2007.

[Trivela *et al.*, 2015] Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos B. Stamou. Optimising resolution-based rewriting algorithms for OWL ontologies. *J. Web Sem.*, 33:30–49, 2015.

Appendix

A Proofs for Section 2

Proposition 3. *Given a Horn-$\mathcal{SHIF}$ (resp. $\mathcal{ELHF}_\bot$) TBox $\mathcal{T}_1$ and ABox signature Σ, one can construct in polynomial time an $\mathcal{ELIHF}_\bot$ (resp. $\mathcal{ELHF}_\bot$) TBox $\mathcal{T}_2$ in normal form such that for every Σ-ABox $\mathcal{A}$,*

1. *$\mathcal{A}$ is consistent with $\mathcal{T}_1$ iff $\mathcal{A}$ is consistent with $\mathcal{T}_2$;*
2. *if $\mathcal{A}$ is consistent with $\mathcal{T}_1$, then for any CQ q that does not use symbols from $\mathsf{sig}(\mathcal{T}_2) \setminus \mathsf{sig}(\mathcal{T}_1)$, we have $\mathsf{cert}(Q_1, \mathcal{A}) = \mathsf{cert}(Q_2, \mathcal{A})$ where $Q_i = (\mathcal{T}_i, \Sigma, q)$.*

Proof. The proof is similar to reductions provided in [Hustadt *et al.*, 2007; Kazakov, 2009]. We sketch the proof for Horn-$\mathcal{SHIF}$. The proof for $\mathcal{ELHF}_\bot$ is similar and omitted.

Assume a $\mathcal{SHIF}$ TBox $\mathcal{T}$ is given. The following rules are used to rewrite $\mathcal{T}$ into an $\mathcal{ELIHF}_\bot$ TBox in normal form. It then only remains to eliminate the transitivity assertions. We assume that the concept names introduced in the rules below are fresh (not in $\mathsf{sig}(\mathcal{T}) \cup \Sigma$):

- If L is of the form $L_1 \sqcap L_2$ and R is not a concept name, then take a fresh concept name A and replace $L \sqsubseteq R$ by $L \sqsubseteq A$ and $A \sqsubseteq R$. If R is a concept name, and either L_1 or L_2 are not concept names, then take fresh concept names A_1, A_2 and replace $L \sqsubseteq R$ by $L_1 \sqsubseteq A_1$, $L_2 \sqsubseteq A_2$ and $A_1 \sqcap A_2 \sqsubseteq R$;
- If L is of the form $L_1 \sqcup L_2$ and R is a concept name, then replace $L \sqsubseteq R$ by $L_1 \sqsubseteq R$ and $L_2 \sqsubseteq R$. Otherwise take a fresh concept name A and replace $L \sqsubseteq R$ by $L \sqsubseteq A$ and $A \sqsubseteq R$;
- If L is of the form $\exists r.L'$ and L' is not a concept name, then take a fresh concept name A' and replace $L \sqsubseteq R$ by $L' \sqsubseteq A'$ and $\exists r.A' \sqsubseteq R$;
- If R is of the form $\neg A$, then replace $L \sqsubseteq R$ by $L \sqcap A \sqsubseteq \bot$;
- If R is of the form $R_1 \sqcap R_2$ and L is not a concept name, then take a fresh concept name A and replace $L \sqsubseteq R$ by $L \sqsubseteq A$ and $A \sqsubseteq R$. Otherwise take fresh concept names A_1, A_2 and replace $L \sqsubseteq R$ by $L \sqsubseteq A_1$, $L \sqsubseteq A_2$, $A_1 \sqsubseteq R_1$, and $A_2 \sqsubseteq R_2$;
- If R is of the form $\neg L' \sqcup R'$, then replace $L \sqsubseteq R$ by $L \sqcap L' \sqsubseteq R'$;
- If R is of the form $\exists r.R'$ and R' is not a concept name, then take a fresh concept name A' and replace $L \sqsubseteq R$ by $L \sqsubseteq \exists r.A'$ and $A' \sqsubseteq R'$;
- If R is of the form $\forall r.R'$, then replace $L \sqsubseteq R$ by $\exists r^-.L \sqsubseteq R$.

The resulting TBox $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$; i.e., it has the following two properties:

- $\mathcal{T}' \models \mathcal{T}$;
- every model $\mathcal{I}$ of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ by appropriately interpreting the fresh concept names.

Now we show how transitivity assertions can be eliminated from $\mathcal{T}'$: for any role r with $\mathcal{T} \models \mathsf{trans}(r)$ and concept name B take a fresh concept name X and add the CIs $\exists r.B \sqsubseteq X$, $\exists r.X \sqsubseteq X$, and $X \sqsubseteq \exists r.B$ to $\mathcal{T}'$. Also remove the transitivity assertions from $\mathcal{T}'$. The resulting TBox, $\mathcal{T}''$, is an $\mathcal{ELIHF}_\bot$ TBox and has the following two properties (we call a role name r *simple relative to* $\mathcal{T}$ if there does not exist a role s with $\mathcal{T} \models \mathsf{trans}(s)$ and $\mathcal{T} \models s \sqsubseteq r$):

- every model of $\mathcal{T}'$ can be extended to a model of $\mathcal{T}''$ by appropriately interpreting the fresh concept names of $\mathcal{T}''$;
- for every model $\mathcal{I}$ of $\mathcal{T}''$ there exists a model $\mathcal{J}$ of $\mathcal{T}'$ which coincides with $\mathcal{I}$ regarding the interpretation of concept names and regarding the interpretation of role names r that are simple relative to $\mathcal{T}$. Moreover, for role names r that are not simple relative to $\mathcal{T}$ we have $r^{\mathcal{J}} \supseteq r^{\mathcal{I}}$.

It follows that $\mathcal{T}''$ is as required since role names that are not simple relative to $\mathcal{T}$ do not occur in any CQs in OMQs. □

We require the following standard characterization of FO-definability. Let $\mathcal{I}$ and $\mathcal{J}$ be interpretations and $\mathbf{a} = a_1, \dots, a_n$ a sequence of individual names. Then $\mathcal{I}$ and $\mathcal{J}$ are called *m-equivalent for Σ and $\mathbf{a}$*, in symbols $\mathcal{I} \equiv_{m,\Sigma,\mathbf{a}} \mathcal{J}$, if $\mathcal{I}$ and $\mathcal{J}$ satisfy the same first-order sentences of quantifier rank $\leq m$ using predicates from Σ and individual constants from $\mathbf{a}$ only. The following characterization of FO-definability is well known and can be proved in a straightforward way.

Lemma 18. *Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ. Then Q is not FO-rewritable iff for all $m > 0$ there are Σ-ABoxes $\mathcal{A}_m$ and $\mathcal{B}_m$ that are consistent with $\mathcal{T}$ and there is $\mathbf{a} \in \mathsf{Ind}(\mathcal{A}_m) \cap \mathsf{Ind}(\mathcal{B}_m)$ such that*

- *$\mathcal{A}_m, \mathcal{T} \models q(\mathbf{a})$ and $\mathcal{B}_m, \mathcal{T} \not\models q(\mathbf{a})$ and*
- *$\mathcal{I}_{\mathcal{A}_m} \equiv_{m,\Sigma,\mathbf{a}} \mathcal{I}_{\mathcal{B}_m}$.*

We use Lemma 18 to prove Theorem 4.

Theorem 4. *Let $\mathcal{L} \in \{\mathcal{ELIHF}_\bot, \mathcal{ELHF}_\bot\}$. Then FO-rewritability in $(\mathcal{L}, CQ)$ can be solved in polynomial time when there is access to oracles for containment in $(\mathcal{L}, \mathcal{Q})$ and for FO-rewritability in $(\mathcal{L}, conCQ)$.*

Proof. Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ in $(\mathcal{L}, \mathrm{CQ})$. Assume $q(\mathbf{x}) = \exists \mathbf{y}.\varphi(\mathbf{x}, \mathbf{y})$. The polynomial time algorithm is as follows:

1. Let $\mathbf{x}_1, \dots, \mathbf{x}_k$ and $\mathbf{y}_1, \dots, \mathbf{y}_k$ be mutually disjoint subsets of $\mathbf{x}$ and $\mathbf{y}$, respectively, such that
$$\Gamma = \{q_1(\mathbf{x}_1) = \exists \mathbf{y}_1 \varphi_1(\mathbf{x}_1, \mathbf{y}_1), \dots, q_k(\mathbf{x}_k) = \exists \mathbf{y}_k \varphi_k(\mathbf{x}_k, \mathbf{y}_k)\}$$
is the set of maximal connected subqueries of q.
2. Obtain Γ' from Γ by removing Boolean CQs q_j that are entailed by the remaining CQs as follows: set $\Gamma_0 = \Gamma$ and assume $\Gamma_0, \dots, \Gamma_j$ have been defined for some $j < k$. Then set $\Gamma_{j+1} := \Gamma_j \setminus \{q_{j+1}\}$ if q_{j+1} is Boolean and
$$\mathcal{A}, \mathcal{T} \models \bigwedge_{q_i \in \Gamma_j \setminus \{q_{j+1}\}} q_i(\mathbf{a}_i) \quad \Rightarrow \quad \mathcal{A}, \mathcal{T} \models q_j$$

holds for all Σ-ABoxes $\mathcal{A}$ and all $\mathbf{a}_i$ in $\mathsf{Ind}(\mathcal{A})$. Otherwise set $\Gamma_{j+1} := \Gamma_j$. Let $\Gamma' := \Gamma_k$. Clearly, Γ' can be computed using an oracle for containment in $(\mathcal{L}, \text{CQ})$.

3. Check FO-rewritability of $(\mathcal{T}, \Sigma, q_i)$ for all $q_i \in \Gamma'$ using an oracle for FO-rewritability in $(\mathcal{L}, \text{conCQ})$.
4. Output 'Q is FO-rewritable' iff all $q_i \in \Gamma'$ are FO-rewritable.

The following claim establishes the correctness of this algorithm.

Claim. Q is FO-rewritable iff all $(\mathcal{T}, \Sigma, q_j)$ with $q_j \in \Gamma'$ are FO-rewritable.

The direction from right to left is trivial. Conversely, assume that some $(\mathcal{T}, \Sigma, q_j)$ with $q_j \in \Gamma'$ is not FO-rewritable. By Lemma 18 we find, for all $m > 0$, Σ-ABoxes $\mathcal{A}_m$ and $\mathcal{B}_m$ that are consistent relative to $\mathcal{T}$ and $\mathbf{a}_j \in \mathsf{Ind}(\mathcal{A}_m) \cap \mathsf{Ind}(\mathcal{B}_m)$ of the same length as $\mathbf{x}_j$ such that

- $\mathcal{A}_m, \mathcal{T} \models q_j(\mathbf{a}_j)$ and $\mathcal{B}_m, \mathcal{T} \not\models q_j(\mathbf{a}_j)$;
- $\mathcal{I}_{\mathcal{A}_m} \equiv_{m,\Sigma,\mathbf{a}_j} \mathcal{I}_{\mathcal{B}_m}$.

Consider the query

$$q'(\mathbf{x}') = \bigwedge_{q_i(\mathbf{x}_i) \in \Gamma' \setminus \{q_j(\mathbf{x}_j)\}} q_i(\mathbf{x}_i).$$

Observe that $q(\mathbf{x}) = q(\mathbf{x}', \mathbf{x}_j)$ and that $q(\mathbf{x})$ is equivalent to $q_j(\mathbf{x}_j) \wedge q'(\mathbf{x}')$. We distinguish two cases.

(1) If q_j is not Boolean, then take some Σ-ABox $\mathcal{A}$ that is consistent relative to $\mathcal{T}$ and with $\mathsf{Ind}(\mathcal{A}) \cap \mathsf{Ind}(\mathcal{A}_m) = \emptyset$ and $\mathsf{Ind}(\mathcal{A}) \cap \mathsf{Ind}(\mathcal{B}_m) = \emptyset$ for all $m > 0$ such that $\mathcal{A}, \mathcal{T} \models q'(\mathbf{a}')$ for some $\mathbf{a}'$ in $\mathsf{Ind}(\mathcal{A})$ of the same length as $\mathbf{x}'$. We obtain for all $m > 0$:

- $\mathcal{A}_n \cup \mathcal{A}, \mathcal{T} \models q(\mathbf{a}', \mathbf{a}_j)$ and $\mathcal{B}_n \cup \mathcal{A} \not\models q(\mathbf{a}', \mathbf{a}_j)$;
- $\mathcal{I}_{\mathcal{A}_n \cup \mathcal{A}} \equiv_{n,\Sigma,\mathbf{a}',\mathbf{a}_j} \mathcal{I}_{\mathcal{B}_n \cup \mathcal{A}}$.

It follows from Lemma 18 that $(\mathcal{T}, \Sigma, q)$ is not FO-rewritable.

(2) If q_j is Boolean, then take some Σ-ABox $\mathcal{A}$ with $\mathsf{Ind}(\mathcal{A}) \cap \mathsf{Ind}(\mathcal{A}_m) = \emptyset$ for all $m > 0$ such that $\mathcal{A}, \mathcal{T} \models q'(\mathbf{a}')$ and $\mathcal{A}, \mathcal{T} \not\models q_j$ for some $\mathbf{a}'$ in $\mathsf{Ind}(\mathcal{A})$ of the same length as $\mathbf{x}'$ (which, since q_j is Boolean, coincides with the length of $\mathbf{x}$). We obtain for all $m > 0$:

- $\mathcal{A}_m \cup \mathcal{A}, \mathcal{T} \models q(\mathbf{a}')$ and $\mathcal{B}_m \cup \mathcal{A} \not\models q(\mathbf{a}')$;
- $\mathcal{I}_{\mathcal{A}_m \cup \mathcal{A}} \equiv_{m,\Sigma,\mathbf{a}'} \mathcal{I}_{\mathcal{B}_m \cup \mathcal{A}}$.

It follows again from Lemma 18 that $(\mathcal{T}, \Sigma, q)$ is not FO-rewritable. □

B Proofs for Section 4

B.1 Preliminary: Role intersections

We extend the DLs $\mathcal{ELIHF}_\bot$ and $\mathcal{ELHF}_\bot$ with intersections of roles that can occur in existential restrictions on the left hand side of concept inclusions. This extension enables us to reduce entailment of tree-shaped CQs to TBox reasoning.

An $\mathcal{ELI}^\cap$ *concept* is an $\mathcal{ELI}$ concept that additionally admits *role intersections* $R = r_1 \cap \cdots \cap r_n$ of roles $r_1, \ldots, r_n$ in existential restrictions. We denote role intersections by R, S, R' etc. An $\mathcal{EL}^\cap$ *concept* is an $\mathcal{EL}$ concept that additionally admits intersections of role names in existential restrictions. An $\mathcal{ELIHF}_\bot^{\cap\text{-lhs}}$ *TBox* is an $\mathcal{ELIHF}_\bot$ TBox in which $\mathcal{ELI}^\cap$ concepts can occur on the left hand side of concept inclusions. Similarly, an $\mathcal{ELHF}_\bot^{\cap\text{-lhs}}$ *TBox* is an $\mathcal{ELHF}_\bot$ TBox in which $\mathcal{EL}^\cap$ concepts can occur on the left hand side of concept inclusions. The semantics of $\mathcal{ELIHF}_\bot^{\cap\text{-lhs}}$ TBoxes is defined by extending the semantics of $\mathcal{ELIHF}_\bot$ in a straightforward manner, where we assume that $R^\mathcal{I} = r_1^\mathcal{I} \cap \cdots \cap r_n^\mathcal{I}$ for any interpretation $\mathcal{I}$ and role inclusion $R = r_1 \cap \cdots \cap r_n$.

The definition of a normal form for TBoxes and Theorem 3 can be easily extended from $\mathcal{ELIHF}_\bot$ to $\mathcal{ELIHF}_\bot^{\cap\text{-lhs}}$: say that an $\mathcal{ELIHF}_\bot^{\cap\text{-lhs}}$ TBox $\mathcal{T}$ is in *normal form* if its concept inclusions take the form

$$A \sqsubseteq \bot \quad A \sqsubseteq \exists r.B \quad \top \sqsubseteq A \quad B_1 \sqcap B_2 \sqsubseteq A \quad \exists R.B \sqsubseteq A$$

with A, B, B_1, B_2 concept names, r a role, and R a role intersection. An analogue of Theorem 3 is formulated and proved in the obvious way. We leave this to the reader.

B.2 Preliminary: Canonical models

We introduce the canonical model $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ of an ABox $\mathcal{A}$ and TBox $\mathcal{T}$ in $\mathcal{ELIHF}_\bot^{\cap\text{-lhs}}$. The main properties of $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ are:

- $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is a model of $\mathcal{A}$ and $\mathcal{T}$;
- for every model $\mathcal{I}$ of $\mathcal{T}$ there exists a homomorphism from $\mathcal{I}_{\mathcal{I},\mathcal{A}}$ to $\mathcal{I}$ that maps each $a \in \mathsf{Ind}(\mathcal{A})$ to itself.

$\mathcal{I}_{\mathcal{A},\mathcal{T}}$ is constructed using a standard chase procedure. We will also introduce a variant of this procedure that constructs, given $\mathcal{A}$ and $\mathcal{T}$, the *completion* ABox $\mathcal{A}^c_\mathcal{T}$ of $\mathcal{A}$ which contains $\mathcal{A}$ and all assertions $A(a)$ and $r(a,b)$ with $a, b \in \mathsf{Ind}(\mathcal{A})$ that are entailed by $\mathcal{A}$ and $\mathcal{T}$. In both cases we assume that $\mathcal{A}$ is consistent with $\mathcal{T}$ and that $\mathcal{T}$ is in normal form.

We start by defining the canonical model $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ of $\mathcal{A}$ and $\mathcal{T}$. It is convenient to use ABox notation when constructing $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ and so we will construct a (possibly infinite) ABox $\mathcal{A}^{\text{can}}_\mathcal{T}$ and define $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ as the interpretation corresponding to $\mathcal{A}^{\text{can}}_\mathcal{T}$.

Thus assume that $\mathcal{A}$ and $\mathcal{T}$ are given. The *full completion sequence of $\mathcal{A}$ w.r.t. $\mathcal{T}$* is the sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \ldots$ defined by setting

$$\begin{aligned} \mathcal{A}_0 \quad = \quad & \mathcal{A} \cup \\ & \{r(a,b) \mid s(a,b) \in \mathcal{A}, \mathcal{T} \models s \sqsubseteq r\} \cup \\ & \{r(a,b) \mid s(b,a) \in \mathcal{A}, \mathcal{T} \models s^- \sqsubseteq r\} \end{aligned}$$

and defining $\mathcal{A}_{i+1}$ to be $\mathcal{A}_i$ extended as follows (recall that we abbreviate $r(a,b)$ by $r^-(b,a)$ and that r ranges over roles):

(i) if $\exists R.B \sqsubseteq A \in \mathcal{T}$ for $R = r_1 \cap \cdots \cap r_n$ and $r_1(a,b), \ldots, r_n(a,b), B(b) \in \mathcal{A}_i$, then add $A(a)$ to $\mathcal{A}_i$;

(ii) if $\top \sqsubseteq A \in \mathcal{T}$ and $a \in \mathsf{Ind}(\mathcal{A}_i)$, then add $A(a)$ to $\mathcal{A}_i$;

(iii) if $B_1 \sqcap B_2 \sqsubseteq A \in \mathcal{T}$ and $B_1(a), B_2(a) \in \mathcal{A}_i$, then add $A(a)$ to $\mathcal{A}_i$;

(iv) if $A \sqsubseteq \exists r.B \in \mathcal{T}$ and $\mathsf{func}(r) \in \mathcal{T}$ and $A(a) \in \mathcal{A}_i$ and there exists b with $r(a,b) \in \mathcal{A}_i$, then add $B(b)$ to $\mathcal{A}_i$;

(v) if $A \sqsubseteq \exists r.B \in \mathcal{T}$ and $\mathsf{func}(r) \notin \mathcal{T}$ and $A(a) \in \mathcal{A}_i$, then take a fresh individual b and add $r(a,b)$ and $B(b)$ to $\mathcal{A}_i$;

(vi) if $r \sqsubseteq s \in \mathcal{T}$ and $r(a,b) \in \mathcal{A}_i$, then add $s(a,b)$ to $\mathcal{A}_i$.

Now let $\mathcal{A}_\mathcal{T}^{\text{can}} = \bigcup_{i \geq 0} \mathcal{A}_i$ and let $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ be the interpretation corresponding to $\mathcal{A}_\mathcal{T}^{\text{can}}$. It is straightforward to prove the following properties of $\mathcal{I}_{\mathcal{A},\mathcal{T}}$.

Lemma 19. *Assume $\mathcal{A}$ is consistent with $\mathcal{T}$ and $\mathcal{T}$ is in normal form. Then*

- *$\mathcal{I}_{\mathcal{T},\mathcal{A}}$ is a model of $\mathcal{A}$ and $\mathcal{T}$;*
- *for every model $\mathcal{I}$ of $\mathcal{T}$ there exists a homomorphism from $\mathcal{I}_{\mathcal{I},\mathcal{A}}$ to $\mathcal{I}$ that maps each $a \in \mathsf{Ind}(\mathcal{A})$ to itself.*

The ABox $\mathcal{A}_\mathcal{T}^{\text{can}}$ can contain additional individuals and can even be infinite. For some purposes it is more convenient to work with the subset $\mathcal{A}_\mathcal{T}^{c}$ of $\mathcal{A}_\mathcal{T}^{\text{can}}$ that only contains those assertions in $\mathcal{A}_\mathcal{T}^{\text{can}}$ that use individual names from $\mathcal{A}$. $\mathcal{A}_\mathcal{T}^{c}$ can be constructed using rules as well. For any individual name a we set

$$\mathcal{A}|_a = \{A(a) \mid A(a) \in \mathcal{A}\}$$

Now consider the rules (i) to (iv) from above and replace the rules (v) and (vi) by the single rule

(vii) if $\mathcal{A}_i|_a, \mathcal{T} \models A(a)$, add $A(a)$ to $\mathcal{A}_i$.

Thus, the *completion sequence of $\mathcal{A}$ w.r.t. $\mathcal{T}$* is the sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \ldots$, where $\mathcal{A}_0$ is as defined above and $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ by applying the rules (i) to (iv) and (vii) to $\mathcal{A}_i$. The proof of the following is straightforward.

Lemma 20. *For all assertions $A(a)$ and $r(a,b)$ with $a, b \in \mathsf{Ind}(\mathcal{A})$:*

- *$\mathcal{A}, \mathcal{T} \models A(a)$ iff $A(a) \in \mathcal{A}_\mathcal{T}^{c}$;*
- *$\mathcal{A}, \mathcal{T} \models r(a,b)$ iff $r(a,b) \in \mathcal{A}_0$ iff $r(a,b) \in \mathcal{A}_\mathcal{T}^{c}$.*

B.3 ABox Unraveling and Proof of Proposition 8

We show that if a CQ is entailed by an ABox $\mathcal{A}$ and TBox $\mathcal{T}$, then it is entailed by an unraveling of $\mathcal{A}$ into a pseudo tree ABox $\mathcal{A}^*$ and the TBox $\mathcal{T}$. The corresponding result has been proved for $\mathcal{ELIF}_\bot$ TBoxes in [Baader *et al.*, 2010] and can be extended to $\mathcal{ELIHF}_\bot$ TBoxes in a straightforward manner. To formulate the result, we need a notion of homomorphisms between ABoxes.

Definition 21. Let $\mathcal{A}$, $\mathcal{B}$ be ABoxes. A mapping $h : \mathsf{Ind}(\mathcal{A}) \to \mathsf{Ind}(\mathcal{B})$ is a *homomorphism* if

- $A(a) \in \mathcal{A}$ implies $A(h(a)) \in \mathcal{B}$ for all $a \in \mathsf{Ind}(\mathcal{A})$;
- $r(a,b) \in \mathcal{A}$ implies $r(h(a), h(b)) \in \mathcal{B}$ for all $a, b \in \mathsf{Ind}(\mathcal{A})$.

The following preservation property of homomorphisms w.r.t. certain answers to CQs is well known.

Lemma 22. *Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from ($\mathcal{ELIHF}_\bot$, CQ), $\mathcal{A}$, $\mathcal{B}$ ABoxes, and h a homomorphism from $\mathcal{A}$ to $\mathcal{B}$ such that every role that is functional in $\mathcal{B}$ is functional in $\mathcal{A}$ as well.*

- *If $\mathcal{B}$ is consistent with $\mathcal{T}$, then $\mathcal{A}$ is consistent with $\mathcal{T}$;*
- *if $\mathcal{A} \models Q(\mathbf{a})$, then $\mathcal{B} \models Q(h(\mathbf{a}))$ for all $\mathbf{a} \subseteq \mathsf{Ind}(\mathcal{A})$.*

Proposition 23. *Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from ($\mathcal{ELIHF}_\bot$, CQ) and let $\mathcal{A}$ be a Σ-ABox that is consistent with $\mathcal{T}$ such that $\mathcal{A} \models Q(\mathbf{a})$. Then there is a pseudo tree Σ-ABox $\mathcal{A}^*$ that is consistent with $\mathcal{T}$, of width at most $|q|$, of outdegree bounded by $|\mathcal{T}|$ and such that $\mathbf{a}$ is in the core of $\mathcal{A}^*$ and the following conditions are satisfied:*

1. *$\mathcal{A}^* \models Q(\mathbf{a})$;*
2. *there is a homomorphism from $\mathcal{A}^*$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$;*
3. *if a role r is functional in $\mathcal{A}$, then r is functional in $\mathcal{A}^*$.*

If $\mathcal{T}$ is an $\mathcal{ELHF}_\bot$ TBox, then there exists a pseudo ditree ABox $\mathcal{A}^*$ with these properties.

Proposition 8. *Let $Q_i = (\mathcal{T}_i, \Sigma, q_i)$, $i \in \{1,2\}$, be OMQs from ($\mathcal{ELIHF}_\bot$, CQ). Then $Q_1 \not\subseteq Q_2$ iff there is a pseudo tree Σ-ABox $\mathcal{A}$ of outdegree at most $|\mathcal{T}_1|$ and width at most $|q_1|$ that is consistent with both $\mathcal{T}_1$ and $\mathcal{T}_2$ and a tuple $\mathbf{a}$ from the core of $\mathcal{A}$ such that $\mathcal{A} \models Q_1(\mathbf{a})$ and $\mathcal{A} \not\models Q_2(\mathbf{a})$.*

If Q_1, Q_2 are from ($\mathcal{ELHF}_\bot$, CQ), then we can find a pseudo ditree ABox with these properties.

Proof. The direction from right to left is trivial. Now assume that $Q_1 \not\subseteq Q_2$. Then there exists a Σ-ABox $\mathcal{A}$ that is consistent with $\mathcal{T}_1$ and $\mathcal{T}_2$ and $\mathbf{a}$ in $\mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A} \models Q_1(\mathbf{a})$ and $\mathcal{A} \not\models Q_2(\mathbf{a})$. By Proposition 23 there exists a pseudo tree Σ-ABox $\mathcal{A}^*$ that is consistent with $\mathcal{T}_1$, of width at most $|q_1|$ and of outdegree bounded by $|\mathcal{T}_1|$ such that $\mathbf{a}$ is in the core of $\mathcal{A}^*$ with

- $\mathcal{A}^* \models Q_1(\mathbf{a})$;
- there is a homomorphism from $\mathcal{A}^*$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$;
- if a role r is functional in $\mathcal{A}$, then r is functional in $\mathcal{A}^*$

It follows from Lemma 22 that $\mathcal{A}^*$ is consistent with $\mathcal{T}_2$ and that $\mathcal{A}^* \not\models Q_2(\mathbf{a})$, as required. □

B.4 Preliminary: Tree-shaped queries

We show how Boolean CQs can be rewritten into a set of tree-shaped CQs and then encoded into $\mathcal{ELIHF}_\bot^{\cap\text{-lhs}}$ TBoxes in such a way that their entailment due to matches in tree-shaped parts of the canonical model is preserved.

For a CQ q we denote by $\mathsf{var}(q)$ the set of variables in q. A CQ q is *weakly tree-shaped* if the undirected graph with nodes $\mathsf{var}(q)$ and edges $\{\{x,x'\} \mid r(x,x') \in q\}$ is acyclic and connected. q is called *weakly ditree-shaped* if the directed graph with nodes $\mathsf{var}(q)$ and edges $\{(x,x') \mid (x,x') \in q\}$ is a tree.

Given a weakly tree-shaped query q we denote by C_q the corresponding $\mathcal{ELI}^\cap$ concept (the obvious $\mathcal{ELI}^\cap$ concept for which for any interpretation $\mathcal{I}$ and any $d \in \Delta^\mathcal{I}$ we have $d \in C^\mathcal{I}$ iff $\mathcal{I} \models q(d)$). If q is a Boolean weakly tree-shaped query, we denote by C_q the $\mathcal{ELI}^\cap$ concept corresponding to an arbitarily chosen query q' that results from q by regarding one of its variables as an answer variable (in what follows it will not matter which variable we choose). Note that if q is a weakly ditree-shaped CQ then we can assume that C_q is an $\mathcal{EL}^\cap$ concept.

Call an interpretation $\mathcal{I}$ *weakly tree-shaped* if the undirected graph with nodes $\Delta^{\mathcal{I}}$ and edges $\{\{d, d'\} \mid (d, d') \in r^{\mathcal{I}}\}$ is acyclic and connected. Call $\mathcal{I}$ *weakly ditree-shaped* if the directed graph with nodes $\Delta^{\mathcal{I}}$ and edges $\{(d, d') \mid (d, d') \in r^{\mathcal{I}}\}$ is a tree. Observe that in the canonical model $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ the interpretation $\mathcal{I}_a$ attached to the individual names $a \in \mathsf{Ind}(\mathcal{A})$ are weakly tree-shaped. Moreover, if $\mathcal{T}$ is an $\mathcal{ELHF}_{\bot}^{\cap\text{-lhs}}$ TBox, then they are weakly ditree-shaped. It follows that in the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ of an $\mathcal{ELIHF}_{\bot}$ TBox $\mathcal{T}$ and pseudo tree ABox $\mathcal{A}$ the only non weakly tree-shaped part is the core of $\mathcal{A}$. Moreover, if $\mathcal{T}$ is a $\mathcal{ELHF}_{\bot}$ TBox then the only non weakly ditree-shaped part of $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ is again the core of $\mathcal{A}$. The following result is straightforward.

Lemma 24. *For any Boolean CQ q there are sets $\mathsf{tree}(q)$ and $\mathsf{dtree}(q)$ of $\mathcal{ELI}^{\cap}$-concepts and, respectively, $\mathcal{EL}^{\cap}$-concepts such that*

1. *$|\mathsf{tree}(q)| \leq 2^{|q|}$ and for any weakly tree-shaped interpretation $\mathcal{I}$, $\mathcal{I} \models q$ iff there exists $C \in \mathsf{tree}(q)$ such that $C^{\mathcal{I}} \neq \emptyset$;*
2. *$|\mathsf{dtree}(q)| \leq 1$ and for any weakly ditree-shaped interpretation $\mathcal{I}$, $\mathcal{I} \models q$ iff there exists $C \in \mathsf{dtree}(q)$ such that $C^{\mathcal{I}} \neq \emptyset$.*

We use simple $\mathcal{ELIHF}_{\bot}^{\cap\text{-lhs}}$ TBoxes to encode entailment of weakly tree-shaped queries. For any set $\mathcal{Q}$ of $\mathcal{ELI}^{\cap}$ concepts denote by $\mathcal{T}_{\mathcal{Q}}$ the $\mathcal{ELIHF}_{\bot}^{\cap\text{-lhs}}$ TBox that is obtained by computing the normal form of

$$\{C \sqsubseteq A_C \mid C \in \mathcal{Q}\},$$

where the A_C are fresh concept names for each $C \in \mathcal{Q}$. A *match* of a CQ $q = \exists \mathbf{x} \varphi(\mathbf{x}, \mathbf{y})$ in an interpretation $\mathcal{I}$ is a mapping π from the variables $\mathbf{x} \cup \mathbf{y}$ of q into $\Delta^{\mathcal{I}}$ such that $\mathcal{I} \models \varphi(\pi(\mathbf{x}, \mathbf{y}))$.

Lemma 25. *Let $Q = (\mathcal{T}, \Sigma, q)$, where $\mathcal{T}$ is an $\mathcal{ELIHF}_{\bot}$ TBox and q is Boolean CQ. Let $\mathcal{A}$ be a pseudo tree Σ-ABox and $\mathcal{T}' = \mathcal{T} \cup \mathcal{T}_{\mathsf{tree}(q)}$. If q has a match in $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ whose range does not intersect with the core of $\mathcal{A}$, then $\mathcal{A}, \mathcal{T}' \models \exists x A_C(x)$ for some $C \in \mathsf{tree}(q)$.*

Moreover, if $\mathcal{T}$ is an $\mathcal{ELHF}_{\bot}$ TBox and $\mathcal{A}$ a pseudo ditree Σ-ABox, then this still holds if $\mathcal{T}'$ is replaced by $\mathcal{T} \cup \mathcal{T}_{\mathsf{dtree}(q)}$ and $\mathsf{tree}(q)$ by $\mathsf{dtree}(q)$.

B.5 Proof of Theorem 9

Theorem 9. *Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from ($\mathcal{ELIHF}_{\bot}$, conCQ). If the arity of q is at least one, then the following conditions are equivalent:*

1. *Q is FO-rewritable;*
2. *there is a $k \geq 0$ such that for all pseudo tree Σ-ABoxes $\mathcal{A}$ that are consistent with $\mathcal{T}$ and of outdegree at most $|\mathcal{T}|$ and width at most $|q|$: if $\mathcal{A} \models Q(\mathbf{a})$ with $\mathbf{a}$ from the core of $\mathcal{A}$, then $\mathcal{A}|_{\leq k} \models Q(\mathbf{a})$;*

If q is Boolean, this equivalence holds with (2.) replaced by

2′. *there is a $k \geq 0$ such that for all pseudo tree Σ-ABoxes $\mathcal{A}$ that are consistent with $\mathcal{T}$ and of outdegree at most $|\mathcal{T}|$ and of width at most $|q|$: if $\mathcal{A} \models Q$, then $\mathcal{A}|_{>0} \models Q$ or $\mathcal{A}|_{\leq k} \models Q$.*

If Q is from ($\mathcal{ELHF}_{\bot}$, conCQ), then the above equivalences hold also when pseudo tree Σ-ABoxes are replaced with pseudo ditree Σ-ABoxes.

Proof. (1) ⇒ (2). Assume φ is an FO rewriting of $Q = (\mathcal{T}, \Sigma, q)$ but (2) does not hold. Let $\mathrm{qr}(\varphi)$ denote the quantifier rank of φ. Consider first the case in which q is Boolean and take $k > 2^{\mathrm{qr}(\varphi)}$ and a Σ-ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$ such that

- $\mathcal{A}, \mathcal{T} \models q$;
- $\mathcal{A}|_{\leq k}, \mathcal{T} \not\models q$;
- $\mathcal{A}|_{>0}, \mathcal{T} \not\models q$.

Let $\mathcal{A}'$ be the disjoint union of $\mathrm{qr}(\varphi)$ many copies of $\mathcal{A}|_{>0}$ and $\mathcal{A}|_{\leq k}$, respectively, and let $\mathcal{A}''$ be the disjoint union of $\mathcal{A}$ and $\mathcal{A}'$. We have

- $\mathcal{A}'', \mathcal{T} \models q$ and $\mathcal{A}', \mathcal{T} \not\models q$ (since q is connected).

Hence $\mathcal{I}_{\mathcal{A}''} \models \varphi$ and $\mathcal{I}_{\mathcal{A}'} \not\models \varphi$. On the other hand, one can easily prove using Ehrenfeucht-Fraïssé games that

- $\mathcal{I}_{\mathcal{A}'} \equiv_{\mathrm{qr}(\varphi),\Sigma,()} \mathcal{I}_{\mathcal{A}''}$.

It follows that $\mathcal{I}_{\mathcal{A}'} \models \varphi$ and we have derived a contradiction.

Now assume that q is not Boolean. Take $k > 2^{\mathrm{qr}(\varphi)}$ and a Σ-ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$ and $\mathbf{a}$ in the core of $\mathcal{A}$ such that

- $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$;
- $\mathcal{A}|_{\leq k}, \mathcal{T} \not\models q(\mathbf{a})$.

Let $\mathcal{A}_0$ be the disjoint union of $\mathrm{qr}(\varphi)$ many copies of $\mathcal{A}$ and $\mathcal{A}|_{\leq k}$, respectively. Now let $\mathcal{A}'$ be the disjoint union $\mathcal{A}_0$ and $\mathcal{A}_{|\leq k}$ and let $\mathcal{A}''$ be the disjoint union of $\mathcal{A}_0$ and $\mathcal{A}$. We have

- $\mathcal{A}'', \mathcal{T} \models q(\mathbf{a})$ and $\mathcal{A}', \mathcal{T} \not\models q(\mathbf{a})$ (since q is connected).

Hence $\mathcal{I}_{\mathcal{A}''} \models \varphi(\mathbf{a})$ and $\mathcal{I}_{\mathcal{A}'} \not\models \varphi(\mathbf{a})$. On the other hand, one can again easily prove using Ehrenfeucht-Fraïssé games that

- $\mathcal{I}_{\mathcal{A}'} \equiv_{\mathrm{qr}(\varphi),\Sigma,\mathbf{a}} \mathcal{I}_{\mathcal{A}''}$.

It follows that $\mathcal{I}_{\mathcal{A}'} \models \varphi(\mathbf{a})$ and we have derived a contradiction.

(2) ⇒ (1). Let k_0 be such that (2) holds. Consider the set Γ of pairs $(\mathcal{A}, \mathbf{c})$ of pseudo tree Σ-ABoxes $\mathcal{A}$ of width at most $|q|$, outdegree at most $|\mathcal{T}|$, and depth at most k_0 that are with to $\mathcal{T}$ and such that $\mathbf{c}$ is in the core of $\mathcal{A}$ and $\mathcal{A}, \mathcal{T} \models q(\mathbf{c})$.

Now regard each $(\mathcal{A}, \mathbf{c}) \in \Gamma$ as a CQ $q_{\mathcal{A},\mathbf{c}}(\mathbf{x})$, where each individual name in $\mathcal{A}$ is viewed as a variable, and $\mathbf{c}$ corresponds to the answer variables $\mathbf{x}$. We will show that

$$\varphi(\mathbf{x}) = \bigvee_{\mathcal{A},\mathbf{c} \in \Gamma} q_{\mathcal{A},\mathbf{c}}(\mathbf{x})$$

is an FO-rewriting of Q.

Assume $\mathcal{A}$ is a Σ ABox that is consistent with $\mathcal{T}$ and $\mathcal{I}_{\mathcal{A}} \models \varphi(\mathbf{a})$. Then $\mathcal{I}_{\mathcal{A}} \models q_{\mathcal{B},\mathbf{c}}(\mathbf{a})$ for some $(\mathcal{B}, \mathbf{c}) \in \Gamma$ and so there is a homomorphism h from $\mathcal{B}$ to $\mathcal{A}$ mapping $\mathbf{c}$ to $\mathbf{a}$. By definition of Γ, it holds that $\mathcal{B}, \mathcal{T} \models q(\mathbf{c})$, and therefore $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$.

Assume that $\mathcal{A}$ is a Σ ABox that is consistent with $\mathcal{T}$ and $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$. By Proposition 23, there is a pseudo tree Σ-ABox $\mathcal{A}^*$ of width at most $|q|$ and outdegree at most $|\mathcal{T}|$ that is consistent with $\mathcal{T}$ such that

- $\mathcal{A}^*, \mathcal{T} \models q(\mathbf{a})$;
- There is a homomorphism h from $\mathcal{A}^*$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$.

Assume first that q is non Boolean. By (2) we have $\mathcal{A}^*|_{\leq k_0}, \mathcal{T} \models q(\mathbf{a})$. Thus $(\mathcal{A}^*|_{\leq k_0}, \mathbf{a}) \in \Gamma$. The homomorphism h (restricted to $\mathcal{A}^*|_{\leq k_0}$) shows that $\mathcal{I}_\mathcal{A} \models \varphi(\mathbf{a})$.

Now assume that q is Boolean. Take a minimal subset $\mathcal{A}'$ of $\mathcal{A}^*$ such that $\mathcal{A}', \mathcal{T} \models q$. $\mathcal{A}'$ is a pseudo tree Σ ABox with some core $\mathcal{A}'_0$. By minimality, $\mathcal{A}'|_{>0}, \mathcal{T} \not\models q$. Thus, by (2) and minimality we have $\mathcal{A}'|_{\leq k_0} = \mathcal{A}'$. Thus $(\mathcal{A}', ()) \in \Gamma$. The homomorphism h (restricted to $\mathcal{A}'$) shows that $\mathcal{I}_\mathcal{A} \models \varphi$.

The proof that for $\mathcal{ELFH}_\bot$ TBoxes it is sufficient to consider pseudo ditree Σ-ABoxes is similar and uses the fact that in Proposition 23 pseudo tree ABoxes can be replaced by pseudo ditree Σ-ABoxes. □

B.6 Proof of Theorem 11

Theorem 11. *Let $\mathcal{T}$ be an $\mathcal{ELIHF}_\bot$ TBox. Then Theorem 9 still holds with the following modifications:*

1. *if q is not Boolean or $\mathcal{T}$ is an $\mathcal{ELHF}_\bot$ TBox, "there is a $k \geq 0$" is replaced with "for $k = |q| + 2^{4(|\mathcal{T}|+|q|)^2}$";*
2. *if q is Boolean, "there is a $k \geq 0$" is replaced with "for $k = |q| + 2^{4(|\mathcal{T}|+2^{|q|})^2}$".*

For the pumping argument, we require some preparation. For an ABox $\mathcal{A}$ and TBox $\mathcal{T}$ we employ the completion sequence $\mathcal{A}_0, \mathcal{A}_1, \ldots$ of $\mathcal{A}$ w.r.t. $\mathcal{T}$ and the completion $\mathcal{A}^c_\mathcal{T}$ defined in the section on canonical models. For an ABox $\mathcal{A}$ and individual a, we set

$$\mathcal{A}|_a = \{A(a) \mid A(a) \in \mathcal{A}, A \in \mathsf{N_C}\}.$$

For a set $\mathcal{X}$ of concepts and an individual u, we set $\mathcal{X}(u) = \{C(u) \mid C \in \mathcal{X}\}$. Let $\mathcal{A}$ be a pseudo tree Σ-ABox and $u \in \mathsf{Ind}(\mathcal{A}_j)$ for some tree of $\mathcal{A}$. Define

$$\mathsf{AT}^\vdash_\mathcal{A}(u) := \{A \in \mathsf{N_C} \mid A(u) \in \mathcal{A}^c_\mathcal{T}\}$$

Let $\mathcal{A}^\downarrow_u$ denote the subtree of $\mathcal{A}_j$ rooted at u, and let $\mathcal{A}^\uparrow_u$ be the ABox obtained from $\mathcal{A}$ by dropping $\mathcal{A}^\downarrow_u$ from $\mathcal{A}$ except for u itself. Define the *transfer sequence* $\mathcal{X}_0, \mathcal{X}_1, \ldots$ *of* $(\mathcal{A}, u)$ *w. r. t.* $\mathcal{T}$ by induction as follows:

- $\mathcal{X}_0 = \mathsf{AT}^\vdash_{\mathcal{A}^0}(u)$, where $\mathcal{A}^0 = \mathcal{A}^\uparrow_u$;
- $\mathcal{X}_1 = \mathsf{AT}^\vdash_{\mathcal{A}^1}(u)$, where $\mathcal{A}^1 = \mathcal{A}^\downarrow_u \cup \mathcal{X}_0(u)$;
- $\mathcal{X}_{2i+2} = \mathsf{AT}^\vdash_{\mathcal{A}^{2i+2}}(u)$, where $\mathcal{A}^{2i+2} = \mathcal{A}^{2i} \cup \mathcal{X}_{2i+1}(u)$, for $i \geq 0$;
- $\mathcal{X}_{2i+1} = \mathsf{AT}^\vdash_{\mathcal{A}^{2i+1}}(u)$, where $\mathcal{A}^{2i+1} = \mathcal{A}^{2i-1} \cup \mathcal{X}_{2i}(u)$, for $i \geq 1$.

The sequence of ABoxes $\mathcal{A}^0, \mathcal{A}^1 \ldots$ defined above is called the *ABox transfer sequence for* $(\mathcal{A}, u)$ *w. r. t.* $\mathcal{T}$.

Lemma 26. *Let $n = |\mathsf{sig}(\mathcal{T})| + 1$. Then $\mathcal{X}_n = \mathcal{X}_m$ for all $m > n$ and $(\mathcal{A}^{n-1})^c_\mathcal{T} \cup (\mathcal{A}^n)^c_\mathcal{T} = \mathcal{A}^c_\mathcal{T}$.*

Proof. By definition, $\mathcal{X}_m \subseteq \mathcal{X}_{m+1}$, for all $m > 0$. Moreover, if $\mathcal{X}_{m+1} = \mathcal{X}_m$ for some $m > 0$ then, by Lemma 20,

- all $\mathcal{A}^{(m+1)+2i}$, $i \geq 0$, coincide;
- all $\mathcal{A}^{(m+2)+2i}$, $i \geq 0$, coincide.

It follows that $\mathcal{X}_{m'} = \mathcal{X}_m$ for all $m' > m$. □

We say that $(\mathcal{A}, a)$ and $(\mathcal{B}, b)$ *coincide locally w.r.t.* $\mathcal{T}$ (in symbols $(\mathcal{A}, a) \sim_\mathcal{T} (\mathcal{B}, b)$):

- $\{A \in \mathsf{sig}(\mathcal{T}) \mid A(a) \in \mathcal{A}\} = \{B \in \mathsf{sig}(\mathcal{T}) \mid B(b) \in \mathcal{B}\}$;
- for every role r with $\mathsf{func}(r) \in \mathcal{T}$: there exists a' with $r(a, a') \in \mathcal{A}^\uparrow_a$ iff there exists b' with $r(b, b') \in \mathcal{B}^\uparrow_b$;
- for every role r with $\mathsf{func}(r) \in \mathcal{T}$: there exists a' with $r(a, a') \in \mathcal{A}^\downarrow_a$ iff there exists b' with $r(b, b') \in \mathcal{B}^\downarrow_b$;

Lemma 27. *Let $\mathcal{A}$ and $\mathcal{B}$ be pseudo tree Σ ABoxes with $a \in \mathsf{Ind}(\mathsf{trees}(\mathcal{A}))$ and $b \in \mathsf{Ind}(\mathsf{trees}(\mathcal{B}))$ such that*

- *$(\mathcal{A}, a)$ and $(\mathcal{B}, b)$ coincide locally w.r.t. $\mathcal{T}$;*
- *the transfer sequence of $(\mathcal{A}, a)$ w.r.t. $\mathcal{T}$ coincides with the transfer sequence of $(\mathcal{B}, b)$ w.r.t. $\mathcal{T}$ and is given by $\mathcal{X}_0, \ldots$.*

Denote by $\mathcal{C}$ the ABox obtained from $\mathcal{A}$ by replacing the subtree $\mathcal{A}^\downarrow_a$ by $\mathcal{B}^\downarrow_b$. Then

- *$\mathcal{X}_0, \ldots$ is also the transfer sequence of $(\mathcal{C}, b)$ w.r.t. $\mathcal{T}$.*
- *Given the ABox transfer sequences $\mathcal{A}^0, \ldots$ and $\mathcal{B}^0, \ldots$ of $(\mathcal{A}, a)$ and $(\mathcal{B}, b)$ w. r. t. $\mathcal{T}$, respectively, the ABox transfer sequence $\mathcal{C}^0, \ldots$ of $(\mathcal{C}, b)$ w. r. t. $\mathcal{T}$ is given by setting $\mathcal{C}^{2i} = \mathcal{A}^{2i}$ and $\mathcal{C}^{2i+1} = \mathcal{B}^{2i+1}$, for $i \geq 0$.*

Proof. Straightforward using Lemma 20. □

Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from $(\mathcal{ELIHF}_\bot, \text{conCQ})$ and $k \geq 0$. A pair $\mathcal{A}$, $\mathbf{a}$ with $\mathcal{A}$ a pseudo tree Σ-ABox and $\mathbf{a}$ a tuple in the core of $\mathcal{A}$ is a *k-entailment witness* for Q if

1. $\mathcal{A}$ is consistent with $\mathcal{T}$;
2. $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$;
3. and
   - q is not Boolean and $\mathcal{A}|_{\leq k}, \mathcal{T} \not\models q(\mathbf{a})$ or
   - q is Boolean, $\mathcal{A}|_{\leq k}, \mathcal{T} \not\models q$ and $\mathcal{A}|_{>0}, \mathcal{T} \not\models q$.

If q is Boolean then we say that $\mathcal{A}$ is a k-entailment witness for Q if $\mathcal{A}$, $()$ is a k-entailment witness for Q.

The following Lemma implies Part 1 of Theorem 11 for queries that are not Boolean.

Lemma 28. *Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from $(\mathcal{ELIHF}_\bot, \text{conCQ})$. If q is not Boolean, then Q is not FO-rewritable iff there exists a k_0-entailment witness for Q of out-degree bounded by $|\mathcal{T}|$ for $k_0 = |q| + 2^{3m^2}$ where $m = |\mathcal{T}|$.*

Proof. The direction ($\Rightarrow$) follows from Theorem 9. Conversely, assume that there is a k_0-entailment witness $\mathcal{A}$, $\mathbf{a}$ for $Q = (\mathcal{T}, \Sigma, q)$. We show that for every $k > k_0$ there exists a k-entailment witness for Q. Then non FO-rewritability of Q follows from Theorem 9.

Assume $\mathcal{A}$, $\mathbf{a}$ is a k-entailment witness for Q for some $k \geq k_0$. It is sufficient to construct a pseudo tree Σ-ABox $\mathcal{B}$ which, together with $\mathbf{a}$ is a k'-entailment witness for Q for some $k' > k$. We may assume w. l. o. g. that $\mathcal{A}$ is minimal

in the sense that, for every individual a from the trees of $\mathcal{A}$ we have $\mathcal{A}^{-a}, \mathcal{T} \not\models q(\mathbf{a})$, where $\mathcal{A}^{-a}$ is obtained from $\mathcal{A}$ by dropping the subtree rooted at a (including a).

Let w be a leaf node in $\mathcal{A}$ of maximal distance from the core of $\mathcal{A}$ and ρ be the root of the tree $\mathcal{A}_i$ of $\mathcal{A}$ containing w. Then the distance of w from ρ is at least $k+1$. Since by Lemma 26 the number of transfer sequences w. r. t. $\mathcal{T}$ does not exceed $2^{|\mathcal{T}|^2}$, on the path from ρ to w there must be at least two individuals u_1 and u_2 with distance at least $|q|$ from ρ such that

(a) $(\mathcal{A}, u_1)$ and $(\mathcal{A}, u_2)$ coincide locally w.r.t. $\mathcal{T}$;

(b) the transfer sequences of $(\mathcal{A}, u_1)$ and $(\mathcal{A}, u_2)$ w.r.t. $\mathcal{T}$ coincide;

(c) the transfer sequences of $(\mathcal{A}^{-w}|, u_1)$ and $(\mathcal{A}^{-w}|, u_2)$ w.r.t. $\mathcal{T}$ coincide.

We may assume that u_1 is between ρ and u_2. Let $\mathcal{B}$ be the ABox obtained from $\mathcal{A}$ by replacing $\mathcal{A}^{\downarrow}_{u_2}$ by $\mathcal{A}^{\downarrow}_{u_1}$ in $\mathcal{A}$. By renaming nodes in $\mathcal{A}^{\downarrow}_{u_1}$, we can assume that the root of the subtree $\mathcal{A}^{\downarrow}_{u_1}$ of $\mathcal{B}$ is denoted by u_2.

We show that $\mathcal{B}$, $\mathbf{a}$ is a $k+1$-entailment witness for $\mathcal{T}$, Σ, and q. To this end we show:

1. $\mathcal{B}$ is consistent relative to $\mathcal{T}$;
2. $\mathcal{B}, \mathcal{T} \models q(\mathbf{a})$;
3. $\mathcal{B}|_{\leq k+1}, \mathcal{T} \not\models q(\mathbf{a})$.

First observe that by (a) $(\mathcal{A}, u_1)$ and $(\mathcal{A}, u_2)$ coincide locally w.r.t. $\mathcal{T}$. Thus, since $\mathcal{A}$ is consistent relative to $\mathcal{T}$, $\mathcal{I}_{\mathcal{B}}$ satisfies the functionality constraints of $\mathcal{T}$. Also, it follows from (b) and Lemma 20 and Lemma 27 that we obtain a canonical model $\mathcal{I}_{\mathcal{B},\mathcal{T}}$ of $\mathcal{B}$ and $\mathcal{T}$ by replacing the subtree-interpretation rooted at u_2 in $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ with the subtree-interpretation rooted at u_1 in $\mathcal{I}_{\mathcal{A},\mathcal{T}}$. Thus, $\mathcal{B}$ is consistent relative to $\mathcal{T}$.

It follows from $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$, the condition that q is connected, and the fact that $\mathcal{A}$ coincides with $\mathcal{B}$ for individuals with distance $\leq |q|$ from the core of $\mathcal{A}$ that $\mathcal{B}, \mathcal{T} \models q(\mathbf{a})$.

It follows from (c), Lemma 20, and Lemma 27 that we obtain a canonical model $\mathcal{I}_{\mathcal{B}^{-w},\mathcal{T}}$ of $\mathcal{B}^{-w}$ and $\mathcal{T}$ by replacing the interpretation $\mathcal{I}_{u_2}$ rooted at u_2 in $\mathcal{I}_{\mathcal{A}^{-w},\mathcal{T}}$ with the interpretation $\mathcal{I}_{u_1}$ rooted at u_1 in $\mathcal{I}_{\mathcal{A}^{-w},\mathcal{T}}$.

Now recall that $\mathcal{A}^{-w}, \mathcal{T} \not\models q(\mathbf{a})$. It follows from the condition that q is connected and the fact that $\mathcal{A}^{-w}$ coincides with $\mathcal{B}^{-w}$ for individuals with distance $\leq |q|$ from the core of $\mathcal{A}$ that $\mathcal{B}^{-w}, \mathcal{T} \not\models q(\mathbf{a})$.

Clearly $\mathcal{B}^{-w} \supseteq \mathcal{B}|_{\leq k+1}$. Thus, $\mathcal{B}|_{\leq k+1}, \mathcal{T} \not\models q(\mathbf{a})$, as required. □

We now consider the case in which q is Boolean. Let $\mathcal{A}$ be a pseudo tree Σ-ABox. In contrast to the non Boolean case, q can have matches in the canonical model $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ that do not hit the core of $\mathcal{A}$. Thus, to ensure that after pumping $\mathcal{A}$, no additional matches of q are introduced we have to consider transfer sequences that are invariant under possible matches of q.

Given a CQ q, we thus consider the additional TBox $\mathcal{T}_{\mathsf{tree}(q)}$ defined above and consider transfer sequence relative to $\mathcal{T} \cup \mathcal{T}_{\mathsf{tree}(q)}$ rather than $\mathcal{T}$ only. Observe that this has the desired effect as the canonical model $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ is weakly tree-shaped except for the individuals in its core.

The following Lemma implies Part 2 of Theorem 11.

Lemma 29. *Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from $(\mathcal{ELIHF}_\bot, \mathit{conCQ})$ If q is Boolean, then Q is not FO-rewritable iff there exists a k_0-entailment witness for Q of outdegree bounded by $|\mathcal{T}|$ for $k_0 = |q| + 2^{4m^2}$ where $m = |\mathcal{T}| + 2^{|q|}$.*

Proof. We modify the proof of Lemma 28. The direction ($\Rightarrow$) follows again from Theorem 9.

Conversely, assume that there is a k_0-entailment witness for Q. We show that for every $k > k_0$ there exists a k-entailment witness for Q.

Assume $\mathcal{A}$ is a k-entailment witness for Q for some $k \geq k_0$. It is sufficient to construct a pseudo tree Σ-ABox $\mathcal{B}$ that is consistent relative to $\mathcal{T}$ and is a k'-entailment witness for Q for some $k' > k$. We may assume w. l. o. g. that $\mathcal{A}$ is minimal in the sense that, for every individual a in any tree of $\mathcal{A}$ we have $\mathcal{A}^{-a}, \mathcal{T} \not\models q$.

Let w be a leaf node in $\mathcal{A}$ of maximal distance from the core of $\mathcal{A}$ and ρ be the root of its tree. Then the distance of w from ρ is at least $k+1$. Since by Lemma 26 the number of transfer sequences w.r.t. $\mathcal{T} \cup \mathcal{T}_{\mathsf{tree}(q)}$ does not exceed $2^{(|\mathcal{T}|+|\mathcal{T}_{\mathsf{tree}(q)}|)^2}$, on the path from ρ to w there must be at least two individuals u_1 and u_2 with distance at least $|q|$ from ρ such that

(a) $(\mathcal{A}, u_1)$ and $(\mathcal{A}, u_2)$ coincide locally;

(b) the transfer sequences of $(\mathcal{A}, u_1)$ and $(\mathcal{A}, u_2)$ w.r.t. $\mathcal{T} \cup \mathcal{T}_{\mathsf{tree}(q)}$ coincide;

(c) the transfer sequences of $(\mathcal{A}^{-w}, u_1)$ and $(\mathcal{A}^{-w}, u_2)$ w.r.t. $\mathcal{T} \cup \mathcal{T}_{\mathsf{tree}(q)}$ coincide.

(d) the transfer sequences of $(\mathcal{A}|_{>0}, u_1)$ and $(\mathcal{A}|_{>0}, u_2)$ w.r.t. $\mathcal{T} \cup \mathcal{T}_{\mathsf{tree}(q)}$ coincide.

We may assume that u_1 is between ρ and u_2. Let $\mathcal{B}$ be the ABox obtained from $\mathcal{A}$ by replacing $\mathcal{A}^{\downarrow}_{u_2}$ by $\mathcal{A}^{\downarrow}_{u_1}$ in $\mathcal{A}$. By renaming nodes in $\mathcal{A}^{\downarrow}_{u_1}$, we can assume that the root of the subtree $\mathcal{A}^{\downarrow}_{u_1}$ of $\mathcal{B}$ is denoted by u_2.

We show that

- $\mathcal{B}$ is consistent with $\mathcal{T}$;
- $\mathcal{B}, \mathcal{T} \models q$;
- $\mathcal{B}|_{>0}, \mathcal{T} \not\models q$;
- $\mathcal{B}|_{\leq k+1}, \mathcal{T} \not\models q$.

The argument for (a) is the same as in the proof of Lemma 28 and omitted.

In what follows we apply the observation that for any Σ-ABox $\mathcal{A}$ one obtains a canonical model of $\mathcal{A}$ and $\mathcal{T}$ from a canonical model of $\mathcal{A}$ and $\mathcal{T} \cup \mathcal{T}_{\mathsf{tree}(q)}$ by taking the reduct to the symbols in $\Sigma \cup \mathcal{T}$. In fact, every canonical model of $\mathcal{A}$ and $\mathcal{T}$ can be obtained in this way.

It follows from Lemma 20 and Lemma 27 and conditions (b), (c), and (d), respectively, that we obtain a canonical model

(b') $\mathcal{I}_{\mathcal{B},\mathcal{T}\cup\mathcal{T}^q}$ of $\mathcal{B}$ and $\mathcal{T} \cup \mathcal{T}_{\mathsf{tree}(q)}$ by replacing the subtree rooted at u_2 in $\mathcal{I}_{\mathcal{A},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}$ with the subtree rooted at u_1 in $\mathcal{I}_{\mathcal{A},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}$;

(c') $\mathcal{I}_{\mathcal{B}^{-w},\mathcal{T}\cup\mathcal{T}^q}$ of $\mathcal{B}^{-w}$ and $\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}$ by replacing the subtree rooted at u_2 in $\mathcal{I}_{\mathcal{A}^{-w},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}$ with the subtree rooted at u_1 in $\mathcal{I}_{\mathcal{A}^{-w},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}$;

(d') $\mathcal{I}_{\mathcal{B}|_{>0},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}$ of $\mathcal{B}|_{>0}$ and $\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}$ by replacing the subtree rooted at u_2 in $\mathcal{I}_{\mathcal{A}|_{>0},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}$ with the subtree rooted at u_1 in $\mathcal{I}_{\mathcal{A}|_{>0},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}$.

To show that $\mathcal{B},\mathcal{T}\models q$, we distinguish two cases: if q has a match in $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ that intersects with the core of $\mathcal{A}$, then q has such a match as well in $\mathcal{I}_{\mathcal{B},\mathcal{T}}$ since $\mathcal{A}$ coincides with $\mathcal{B}$ for individuals with distance $\leq|q|$ from the core of $\mathcal{A}$. If q does not have such match, then $\mathcal{A},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}\models\exists x\,A_C(x)$ for some $C\in\mathsf{trees}(q)$. But then $A_C^{\mathcal{I}_{\mathcal{A},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}}\neq\emptyset$ and so $A_C^{\mathcal{I}_{\mathcal{B},\mathcal{T}\cup\mathcal{T}_{\mathsf{tree}(q)}}}\neq\emptyset$. Thus $\mathcal{B},\mathcal{T}\models q$.

$\mathcal{B}|_{>0},\mathcal{T}\not\models q$ follows from (d') and the fact that $\mathcal{A}|_{>0},\mathcal{T}\not\models q$.

To show $\mathcal{B}|_{\leq k+1},\mathcal{T}\not\models q$ it is sufficient to show that $\mathcal{B}^{-w},\mathcal{T}\not\models q$. But this follows from (c') and the fact that $\mathcal{A}^{-w},\mathcal{T}\not\models q$. □

It remains to prove the claim of Theorem 11 for $\mathcal{ELHF}_\bot$ TBoxes and Boolean queries. In this case, since one can work with pseudo ditree Σ-ABoxes rather than arbitrary pseudo tree Σ-ABoxes one can employ the linear size TBox $\mathcal{T}_{\mathsf{dtree}(q)}$ rather than the possibly exponential size TBox $\mathcal{T}_{\mathsf{tree}(q)}$. The remaining part of the proof is exactly the same as before and so one obtains:

Lemma 30. *Let $Q=(\mathcal{T},\Sigma,q)$ be an OMQ from $(\mathcal{ELHF}_\bot,\mathit{conCQ})$. If q is Boolean, then Q is not FO-rewritable iff there exists a k_0-entailment witness for Q of outdegree bounded by $|\mathcal{T}|$ for $k_0=|q|+2^{4m^2}$ where $m=|\mathcal{T}|+|q|$.*

C Proofs for Section 5

C.1 Preliminary: Tree Automata

We introduce two-way alternating parity automata on finite trees (TWAPAs). We assume w.l.o.g. that all nodes in an m-ary tree are from $\{1,\dots,m\}^*$. For any set X, let $\mathcal{B}^+(X)$ denote the set of all positive Boolean formulas over X, i.e., formulas built using conjunction and disjunction over the elements of X used as propositional variables, and where the special formulas true and false are allowed as well. An *infinite path* P of a tree T is a prefix-closed set $P\subseteq T$ such that for every $i\geq 0$, there is a unique $x\in P$ with $|x|=i$.

Definition 31 (TWAPA)**.** A *two-way alternating parity automaton (TWAPA) on finite m-ary trees* is a tuple $\mathfrak{A}=(S,\Gamma,\delta,s_0,c)$ where S is a finite set of *states*, Γ is a finite alphabet, $\delta:S\times\Gamma\to\mathcal{B}^+(\mathsf{tran}(\mathfrak{A}))$ is the *transition function* with $\mathsf{tran}(\mathfrak{A})=\{\langle i\rangle s,\ [i]s\mid -1\leq i\leq m \text{ and } s\in S\}$ the set of *transitions* of $\mathfrak{A}$, $s_0\in S$ is the *initial state*, and $c:S\to\mathbb{N}$ is the *parity condition* that assigns to each state a *priority*.

Intuitively, a transition $\langle i\rangle s$ with $i>0$ means that a copy of the automaton in state s is sent to the i-th successor of the current node, which is then required to exist. Similarly, $\langle 0\rangle s$ means that the automaton stays at the current node and switches to state s, and $\langle -1\rangle s$ indicates moving to the predecessor of the current node, which is then required to exist. Transitions $[i]s$ mean that a copy of the automaton in state s is sent to the relevant successor if that successor exists (which is not required).[1]

Definition 32 (Run, Acceptance)**.** A *run* of a TWAPA $\mathfrak{A}=(S,\Gamma,\delta,s_0,c)$ on a finite Γ-labeled tree (T,L) is a $T\times S$-labeled tree (T_r,r) such that the following conditions are satisfied:

1. $r(\varepsilon)=(\varepsilon,s_0)$;
2. if $y\in T_r$, $r(y)=(x,s)$, and $\delta(s,L(x))=\varphi$, then there is a (possibly empty) set $S\subseteq\mathsf{tran}(\mathfrak{A})$ such that S (viewed as a propositional valuation) satisfies φ as well as the following conditions:
   (a) if $\langle i\rangle s'\in S$, then $x\cdot i$ is defined and there is a node $y\cdot j\in T_r$ such that $r(y\cdot j)=(x\cdot i,s')$;
   (b) if $[i]s'\in S$ and $x\cdot i$ is defined and in T, then there is a node $y\cdot j\in T_r$ such that $r(y\cdot j)=(x\cdot i,s')$.

We say that (T_r,r) is *accepting* if on all infinite paths $\varepsilon=y_1y_2\cdots$ of T_r, the maximum priority that appears infinitely often is even. A finite Γ-labeled tree (T,L) is *accepted* by $\mathfrak{A}$ if there is an accepting run of $\mathfrak{A}$ on (T,L). We use $L(\mathfrak{A})$ to denote the set of all finite Γ-labeled tree accepted by $\mathfrak{A}$.

It is known (and easy to see) that TWAPAs are closed under complementation and intersection, and that these constructions involve only a polynomial blowup. It is also known that their emptiness problem can be solved in time single exponential in the number of states and polynomial in all other components of the automaton. In what follows, we shall generally only explicitly analyze the number of states of a TWAPA, but only implicitly take care that all other components are of the allowed size for the complexity result that we aim to obtain.

Lemma 12. *Let $\mathcal{L}\in\{\mathcal{ELIHF}_\bot,\mathcal{ELHF}_\bot\}$. Then*

1. *FO-rewritability in $(\mathcal{L},\mathit{conCQ})$ can be reduced in polytime to FO-rewritability in $(\mathcal{L},\mathit{conBCQ})$;*
2. *Containment in $(\mathcal{L},\mathit{CQ})$ can be reduced in polytime to containment in $(\mathcal{L},\mathit{BCQ})$.*

Proof. (1.) Consider an OMQ $Q=(\mathcal{T},\Sigma,q(\mathbf{x}))$, where $\mathbf{x}=x_1,\dots,x_n$. Take fresh concept names $A_1,\dots,A_n$ and let $\Sigma'=\Sigma\cup\{A_1,\dots,A_n\}$. Denote by $q'(\mathbf{x})$ the result of adding the conjuncts $A_j(x_j)$ to $q(\mathbf{x})$ for $j\in\{1,\dots,n\}$. One can show that $Q'=(\mathcal{T},\Sigma',\exists\mathbf{x}q'(\mathbf{x}))$ is FO-rewritable iff Q is FO-rewritable. In fact, if $\varphi(\mathbf{x})$ is an FO-rewriting of Q, then obtain $\psi(\mathbf{x})$ from $\varphi(\mathbf{x})$ by adding the conjuncts $A_j(x)$ for $j\in\{1,\dots,n\}$ to $\varphi(\mathbf{x})$. Then $\exists\mathbf{x}\psi(\mathbf{x})$ is an FO-rewriting of Q'.

(2.) Let $Q_i=(\mathcal{T}_i,\Sigma,q_i(\mathbf{x}))$ be OMQs for $i=1,2$. We form the Boolean OMQs Q_i' in the same way as above. Then $Q_1\subseteq Q_2$ iff $Q_1'\subseteq Q_2'$, as required. □

[1] In our automata constructions, we will explicitly use only transitions of the form $\langle i\rangle s$. The dual transitions are needed for closure of TWAPAs under complement.

C.2 Preliminary: Forest Decompositions

Before proving Proposition 13, we carefully analyse query matches in canonical models of pseudo tree ABoxes. We start with the case where the TBox is formulated in $\mathcal{ELIHF}_\bot$ and afterwards consider $\mathcal{ELHF}_\bot$.

Let q be a connected CQ. We use $\mathsf{id}(q)$ to denote the set of all queries that can be obtained from q by identifying variables. A *forest decomposition* of q is a tuple $F = (q_{\mathsf{core}}, q_1, x_1, \dots, q_k, x_k, \mu)$ where $(q_{\mathsf{core}}, q_1, \dots, q_k)$ is a partition of (the atoms of) a query from $\mathsf{id}(q)$, $x_1, \dots, x_k$ are variables from q_{core}, and μ is a mapping from $\mathsf{Var}(q_{\mathsf{core}})$ to $\mathsf{Ind}_{\mathsf{core}}$ such that the following conditions are satisfied for $1 \leq i, j \leq k$;

1. q_{core} is non-empty;
2. q_i is weakly tree-shaped with root x_i;
3. $\mathsf{Var}(q_i) \cap \mathsf{Var}(q_{\mathsf{core}}) = \{x_i\}$;
4. $\mathsf{Var}(q_i) \cap \mathsf{Var}(q_j) \subseteq \mathsf{Var}(q_{\mathsf{core}})$ if $i \neq j$;
5. q_i contains no atom $A(x_i)$;
6. x_i has a single successor in q_i.

With $\mathsf{fdec}(q)$, we denote the set of all forest decompositions of q. The following lemma shows how certain matches of q in the canonical models of pseudo tree ABoxes give rise to forest decompositions of q.

Lemma 33. *Let $\mathcal{T}$ be an $\mathcal{ELIHF}_\bot$ TBox, $\mathcal{A}$ a pseudo tree ABox, and q a Boolean connected CQ. Then the following are equivalent:*

1. *there is a match π of q in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that at least one individual from the core of $\mathcal{A}$ is in the range of π;*
2. *there is a forest decomposition $F = (q_{\mathsf{core}}, q_1, x_1, \dots, q_k, x_k, \mu)$ of q such that*
   - *μ is a match for q_{core} in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ whose range consists solely of core individuals from $\mathcal{A}$;*
   - *for $1 \leq i \leq k$, there is a match π_i for q_i in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi_i(x_i) = \mu(x_i)$.*

Lemma 33 is a minor variation of similar lemmas proved e.g. in [Lutz, 2008]; proof details are omitted.

It can be verified that there is a polynomial p such that for every connected CQ q, the number of forest decompositions of q is bounded by $2^{p(|q|)}$.

Now for the case of $\mathcal{ELHF}_\bot$. We say that a CQ q' is obtained from a CQ q *by fork elimination* if q' is obtained from q by selecting two atoms $r(x, z), s(y, z)$ and identifying the variables x and y. We call q' a *fork rewriting* of q if q' can be obtained from q by repeated (but not necessarily exhaustive) fork elimination. We say that q' is a *maximal fork rewriting* of q if it is a fork rewriting and no further fork elimination is possible. A *directed forest decomposition* is defined like a forest decomposition except that

1. $(q_{\mathsf{core}}, q_1, \dots, q_k)$ is a partition of (the atoms of) a fork rewriting of q, instead of a query from $\mathsf{id}(q)$;
2. the queries $q_1, \dots, q_k$ are required to be weakly ditree-shaped.

We now establish a strengthened version of Lemma 33 for $\mathcal{ELHF}_\bot$ TBoxes.

Lemma 34. *When $\mathcal{T}$ is an $\mathcal{ELHF}_\bot$ TBox, then the equivalence in Lemma 33 is true when forest decompositions are replaced with directed forest decompositions.*

C.3 Preliminary: Derivation Trees

We characterize entailment of AQs in terms of derivation trees. Fix an $\mathcal{ELIHF}_\bot^{\cap\text{-}\mathsf{lhs}}$ TBox $\mathcal{T}$ in normal form and an ABox $\mathcal{A}$. A *derivation tree* for an assertion $A_0(a_0)$ in $\mathcal{A}$ with $A_0 \in \mathsf{N_C} \cup \{\bot\}$ is a finite $\mathsf{Ind}(\mathcal{A}) \times (\mathsf{N_C} \cup \{\bot\})$-labeled tree (T, V) that satisfies the following conditions:

1. $V(\varepsilon) = (a_0, A_0)$;
2. if $V(x) = (a, A)$ and neither $A(a) \notin \mathcal{A}$ nor $\top \sqsubseteq A \in \mathcal{T}$, then one of the following holds:
   - x has successors $y_1, \dots, y_k$, $k \geq 1$ with $V(y_i) = (a, A_i)$ for $1 \leq i \leq k$ and $\mathcal{T} \models A_1 \sqcap \dots \sqcap A_k \sqsubseteq A$;
   - x has a single successor y with $V(y) = (b, B)$ and there is an $\exists R.B \sqsubseteq A \in \mathcal{T}$ and an $R'(a, b) \in \mathcal{A}$ such that $\mathcal{T} \models R' \sqsubseteq R$;
   - x has a single successor y with $V(y) = (b, B)$ and there is a $B \sqsubseteq \exists r.A \in \mathcal{T}$ such that $r(b, a) \in \mathcal{A}$ and $\mathsf{func}(r) \in \mathcal{T}$.

Note that the first item of Point 2 above requires $\mathcal{T} \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq A$ instead of $A_1 \sqcap A_2 \sqsubseteq A \in \mathcal{T}$ to 'shortcut' anonymous parts of the canonical model. In fact, the derivation of A from $A_1 \sqcap \dots \sqcap A_n$ by $\mathcal{T}$ can involve the introduction of anonymous elements.

We call a TBox $\mathcal{T}$ *satisfiable* if it has a model. The main property of derivation trees is the following.

Lemma 35.

1. *$\mathcal{A}, \mathcal{T} \models A(a)$ iff $\mathcal{A}$ is inconsistent with $\mathcal{T}$ or there is a derivation tree for $A(a)$ in $\mathcal{A}$, for all assertions $A(a)$ with $A \in \mathsf{N_C}$ and $a \in \mathsf{Ind}(\mathcal{A})$;*
2. *$\mathcal{A}$ is inconsistent with $\mathcal{T}$ iff $\mathcal{T}$ is unsatisfiable, there is a derivation tree for $\bot(a)$ in $\mathcal{A}$ for some $a \in \mathsf{Ind}(\mathcal{A})$, or $\mathcal{A}$ violates a functionality assertion in $\mathcal{T}$.*

The proof is a straightforward extension of an analogous result for $\mathcal{ELI}_\bot$ in [Bienvenu *et al.*, 2013]. Details are omitted.

C.4 Proof of Proposition 13

We next give the main automaton construction underlying our upper complexity bounds, stated as Proposition 13 in the main paper. For convenience, we repeat the proposition here.

Proposition 13. *For every OMQ $Q = (\mathcal{T}, \Sigma, q)$ from $(\mathcal{ELIHF}_\bot, \mathit{BCQ})$, there is a TWAPA*

1. *$\mathfrak{A}_Q$ that accepts a $(|\mathcal{T}| \cdot |q|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree (T, L) iff it is proper, $\mathcal{A}_{(T,L)}$ is consistent with $\mathcal{T}$, and $\mathcal{A}_{(T,L)} \models Q$;*

   $\mathfrak{A}_Q$ has at most $2^{p(|q| + \log(|\mathcal{T}|))}$ states, and at most $p(|q| + |\mathcal{T}|)$ states if $\mathcal{T}$ is an $\mathcal{ELHF}_\bot$ TBox, p a polynomial.
2. *$\mathfrak{A}_\mathcal{T}$ that accepts a $(|\mathcal{T}| \cdot |q|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree (T, L) iff it is proper and $\mathcal{A}_{(T,L)}$ is consistent with $\mathcal{T}$.*

   $\mathfrak{A}_\mathcal{T}$ has at most $p(|\mathcal{T}|)$ states, p a polynomial.

We can construct $\mathfrak{A}_Q$ and $\mathfrak{A}_\mathcal{T}$ in time polynomial in their size.

We start with proving Point 1 of Proposition 13. Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from $(\mathcal{ELIHF}_\bot, \text{BCQ})$. The automaton $\mathfrak{A}_Q$ is the intersection of two automata $\mathfrak{A}_{Q,1}$ and $\mathfrak{A}_{Q,2}$ and the automaton $\mathfrak{A}_\mathcal{T}$ from Point 2 of Proposition 13. All of them run on $(|\mathcal{T}| \cdot |q|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled trees. The first automaton $\mathfrak{A}_{Q,1}$ accepts the input tree (T, L) iff it is proper. This automaton is very simple to construct and its number of states is polynomial in $|\mathcal{T}|$ and independent of q; we omit details. The second automaton $\mathfrak{A}_{Q,2}$ accepts (T, L) iff $\mathcal{A}_{(T,L)} \models Q$, provided that $\mathcal{A}_{(T,L)}$ is consistent with $\mathcal{T}$.

Before constructing $\mathfrak{A}_{Q,2}$, we first extend the TBox $\mathcal{T}$ as follows. Let $q_1, \dots, q_\ell$ be the maximal connected components of the BCQ q from Q. We use $\mathcal{Q}$ to denote the set of queries that contains

- all queries from $\mathsf{id}(q_1) \cup \dots \cup \mathsf{id}(q_\ell)$ which are weakly tree-shaped;
- the queries $p_1, \dots, p_k$ from any forest decomposition $(p_\mathsf{core}, p_1, x_1, \dots, p_k, x_k, \mu)$ in $\mathsf{fdec}(q_1) \cup \dots \cup \mathsf{fdec}(q_\ell)$.

Each query in $q' \in \mathcal{Q}$ can be viewed as an $\mathcal{ELI}^\cap$-concept $C_{q'}$. We add to $\mathcal{T}$ the inclusion $C_{q'} \sqsubseteq A_{q'}$ for each $q' \in \mathcal{Q}$, and convert to normal form. Call the resulting TBox $\mathcal{T}^+$. The following lemma will guide the construction of the automaton $\mathfrak{A}_{Q,2}$.

Lemma 36. *Let $\mathcal{A}$ be a pseudo tree ABox that is consistent with $\mathcal{T}$. Then $\mathcal{A}, \mathcal{T} \models q$ iff for all maximal connected components q_j of q, one of the following properties is satisfied:*

1. *there is a forest decomposition $F = (q_\mathsf{core}, q_1, x_1, \dots, q_k, x_k, \mu)$ of q_j such that*
   - *μ is a match for q_core in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ whose range consists solely of core individuals from $\mathcal{A}$;*
   - *for $1 \le i \le k$, there is a match π for q_i in $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ such that $\pi(x_i) = \mu(x_i)$;*
2. *there is a query $q' \in \mathsf{id}(q_j)$ that is weakly tree-shaped and an $a \in \mathsf{Ind}(\mathcal{A})$ that is not from the core part of $\mathcal{A}$ and satisfies $\mathcal{A}, \mathcal{T}^+ \models A_{q'}(a)$;*
3. *there is an $a \in \mathsf{Ind}(\mathcal{A})$ and a set S of concept names from $\mathcal{T}$ such that $a \in A^{\mathcal{I}_{\mathcal{A},\mathcal{T}}}$ for all $A \in S$ and $\mathcal{A}_S, \mathcal{T} \models q_j$, where $\mathcal{A}_S = \{A(a) \mid A \in S\}$.*

Proof. (sketch) $\mathcal{A}, \mathcal{T} \models q$ is witnessed by a match for every maximal connected component q_j of q into the canonical model $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ of $\mathcal{A}$ and $\mathcal{T}$. We distinguish three kinds of such matches: (i) Matches that involve a core individual of $\mathcal{A}$: As q_j is connected, by Lemma 33 we directly obtain Point 1. (ii) Matches that involve a tree individual a of $\mathcal{A}$ but no core individual define a weakly tree-shaped query q' that results from identifying variables in q_j. The root of q' is mapped to a, and as $C_{q'} \sqsubseteq A_{q'} \in \mathcal{T}^+$, it follows that $\mathcal{A}, \mathcal{T}^+ \models A_{q'}(a)$. (iii) Matches that do not involve any ABox individuals: Consider the ABox individual a that is the root of the anonymous tree q_j is mapped to. As $\mathcal{T}^+$ is in normal form, it is easy to prove using canonical models that there is a set S of concept names from $\mathcal{T}$ such that $a \in A^{\mathcal{I}_{\mathcal{A},\mathcal{T}}}$ for all $A \in S$ and $\mathcal{A}_S, \mathcal{T} \models q_j$. □

We use $\mathsf{CN}(\mathcal{T}^+)$ to denote the set of all concept names in $\mathcal{T}^+$ (and likewise for $\mathsf{CN}(\mathcal{T})$) and $\mathsf{rol}(\mathcal{T}^+)$ to denote the set of all role intersections in $\mathcal{T}^+$. Define the TWAPA $\mathfrak{A}_{Q,2} = (S, \Sigma, \delta, s_0, c)$ by setting

$$S = \{s_0, s^j_\mathsf{core}, s^j_\mathsf{tree}, s^j_\mathsf{anon} \mid 1 \le j \le \ell\} \uplus \{s_{A,a},\ s_A,\ s_{A,R,a},\ s_{A,R},\ s_{A,R,\uparrow} \mid a \in \mathsf{Ind}_\mathsf{core},\ R \in \mathsf{rol}(\mathcal{T}^+),\ A \in \mathsf{CN}(\mathcal{T}^+)\}$$

and $c(s) = 1$ for all $s \in S$ (i. e., exactly the finite runs are accepting). We introduce the transition function δ in several steps and provide some explanation along the way. Start by setting for all $\rho \in \Sigma_\varepsilon$

- $\delta(s_0, \rho) = \bigwedge_{j=1..\ell} \langle 0 \rangle s^j_\mathsf{core} \vee \langle 0 \rangle s^j_\mathsf{tree} \vee \langle 0 \rangle s^j_\mathsf{anon}$,

which distinguishes the three cases in Lemma 36, for each connected component of q. Next put for all $\rho \in \Sigma_\varepsilon$, $\nu \in \Sigma_N$, and $1 \le j \le \ell$:

- $\delta(s^j_\mathsf{core}, \rho) = \bigvee_{\substack{(p_\mathsf{core}, p_1, x_1, \dots, p_k, x_k, \mu) \in \mathsf{fdec}(q_j) \\ \text{with } \mu \text{ a homomorphism from } p_\mathsf{core} \text{ to } \rho}} \bigwedge_{i \in 1..k} \langle 0 \rangle s_{A_{p_i}, \mu(x_i)}$;
- $\delta(s^j_\mathsf{tree}, \rho) = \bigvee_{i \in 1..m} \langle i \rangle s^j_\mathsf{tree}$;
- $\delta(s^j_\mathsf{tree}, \nu) = \bigvee_{q' \in \mathsf{id}(q_j)} \langle 0 \rangle s_{A_{q'(x)}} \vee \bigvee_{i \in 1..m} \langle i \rangle s^j_\mathsf{tree}$;
- $\delta(s^j_\mathsf{anon}, \rho) = \bigvee_{a \in \mathsf{Ind}(\rho)} \bigvee_{\substack{S \subseteq \mathsf{CN}(\mathcal{T}) \\ \mathcal{A}_S, \mathcal{T} \models q_j}} \Big(\bigwedge_{A \in S} \langle 0 \rangle s^j_{A,a} \Big) \vee \bigvee_{i \in 1..m} \langle i \rangle s^j_\mathsf{anon}$;
- $\delta(s^j_\mathsf{anon}, \nu) = \bigvee_{\substack{S \subseteq \mathsf{CN}(\mathcal{T}) \\ \mathcal{A}_S, \mathcal{T} \models q_j}} \Big(\bigwedge_{A \in S} \langle 0 \rangle s^j_A \Big) \vee \bigvee_{i \in 1..m} \langle i \rangle s^j_\mathsf{anon}$.

The first line selects a forest decomposition as in Point 1 of Lemma 36; lines two and three select an ABox individual in a tree component of the pseudo tree ABox $\mathcal{A}_{(T,L)}$ as in Point 2 of that lemma; and lines four and five select an individual from $\mathcal{A}_{(T,L)}$ as in Point 3, which can be either in the core part or in a tree part. It remains to implement the proof obligations expressed by states of the form $s_{A,a}$ and s_A. The former indicates that the concept name A is made true in the canonical model by the core individual a and the latter that A is made true in the canonical model by the tree individual that corresponds to the current point of the input tree. We make sure that these obligations are satisfied by checking the existence of corresponding derivation trees according to Lemma 35. We start with doing this for the core part of pseudo tree ABoxes. Set for all $\rho \in \Sigma_\varepsilon$ and all $\nu \in \Sigma_N$:

- $\delta(s_{A,a}, \rho) = \mathsf{true}$ if $A(a) \in \rho$ or $a \in \mathsf{Ind}(\rho)$ and $\top \sqsubseteq A \in \mathcal{T}^+$;
- $\delta(s_{A,a}, \rho) = \bigvee_{\mathcal{T}^+ \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq A} \big(\langle 0 \rangle s_{A_1,a} \wedge \dots \wedge \langle 0 \rangle s_{A_n,a} \big) \vee$

$$\bigvee_{\exists R.B \sqsubseteq A \in \mathcal{T}^+} \Big(\bigvee_{R'(a,b)\in\rho \text{ with } \mathcal{T}^+ \models R' \sqsubseteq R} \langle 0\rangle s_{B,b} \vee \bigvee_{i\in 1..m} \langle i\rangle s_{B,R,a} \Big) \vee \bigvee_{B \sqsubseteq \exists r.A \in \mathcal{T}^+,\ \mathsf{func}(r)\in\mathcal{T}^+} \Big(\bigvee_{r(b,a)\in\rho} \langle 0\rangle s_{B,b} \vee \bigvee_{i\in 1..m} \langle i\rangle s_{B,r^-,a} \Big)$$

if $a \in \mathsf{Ind}(\rho)$ and $A(a) \notin \rho$;

- $\delta(s_{A,R,a}, \nu) = \langle 0\rangle s_A$ if $a \in \nu$ and there is an $R' \in \nu$ with $\mathcal{T}^+ \models R' \sqsubseteq R$.

It should be obvious how the above transitions verify the existence of a derivation tree. Note that the tree must be finite since runs are required to be finite. We now deal with proof obligations in the trees of pseudo tree ABoxes. Set for all $\rho \in \Sigma_\varepsilon$ and all $\nu \in \Sigma_N$:

- $\delta(s_A, \nu) =$ true for all $s_A \in S$ with $A \in \nu$ or $\top \sqsubseteq A \in \mathcal{T}^+$;
- $$\delta(s_A,\nu) = \bigvee_{\mathcal{T}^+ \models A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A} \big(\langle 0\rangle s_{A_1} \wedge \cdots \wedge \langle 0\rangle s_{A_n}\big) \vee \bigvee_{\exists R.B \sqsubseteq A \in \mathcal{T}^+} \Big(\langle 0\rangle s_{B,R^-,\uparrow} \vee \bigvee_{i\in 1..m} \langle i\rangle s_{B,R}\Big) \vee \bigvee_{B \sqsubseteq \exists r.A \in \mathcal{T}^+,\ \mathsf{func}(r)\in\mathcal{T}^+} \Big(\langle 0\rangle s_{B,r,\uparrow} \vee \bigvee_{i\in 1..m} \langle i\rangle s_{B,r^-}\Big)$$

  for all $A \in \mathsf{CN}(\mathcal{T}^+)$ with $A \notin \nu$;
- $\delta(s_{A,R,\uparrow}, \nu) = \langle -1\rangle s_A$ if there is an $R' \in \nu$ with $\mathcal{T}^+ \models R' \sqsubseteq R$ and $\nu \cap \mathsf{Ind}_{\mathsf{core}} = \emptyset$;
- $\delta(s_{A,R,\uparrow}, \nu) = \langle -1\rangle s_{A,a}$ if there is an $R' \in \nu$ with $\mathcal{T}^+ \models R' \sqsubseteq R$ and $a \in \nu$;
- $\delta(s_{A,R}, \nu) = \langle 0\rangle s_A$ if there is an $R' \in \nu$ with $\mathcal{T}^+ \models R' \sqsubseteq R$.

We define $\delta(s, \alpha) = \mathsf{false}$ for all $s \in S, \alpha \in \Sigma_\varepsilon \cup \Sigma_N$ not covered above. Using the intuitions above and Lemmas 33 and 35, one can prove the following.

Lemma 37. *Let (T, L) be a $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree that is proper and such that $\mathcal{A}_{(T,L)}$ is consistent with $\mathcal{T}$. Then $\mathfrak{A}_{Q,2}$ accepts (T, L) iff $\mathcal{A}_{(T,L)} \models Q$.*

By construction and Lemma 37, the overall TWAPA $\mathfrak{A}_Q$ accepts the tree language described in Proposition 13 and it remains to analyse its size.

First assume that $\mathcal{T}$ is formulated in $\mathcal{ELIHF}_\bot$. Using the bound on the number of forest decompositions from Section C.2, it can be verified that the size of the extended TBox $\mathcal{T}^+$ is at most $2^{p(|q|+\log(|\mathcal{T}|))}$ for some polynomial p and thus the same is true for the number of states of the TWAPA $\mathfrak{A}_{Q,2}$. Since the number of states of $\mathfrak{A}_{Q,1}$ is polynomial in the size of $\mathcal{T}$ and independent of q and by the size bounds for $\mathfrak{A}_\mathcal{T}$ given in Point 2 of Proposition 13 (and since intersection blows up TWAPAs only polynomially), the bound of at most $2^{p(|q|+\log(|\mathcal{T}|))}$ states also applies to the overall TWAPA $\mathfrak{A}_Q$.

Now for case when $\mathcal{T}$ is formulated in $\mathcal{ELHF}_\bot$. By Lemma 34, we can then replace forest decompositions with directed forest decompositions in the construction of $\mathfrak{A}_Q$. Moreover, Point 2 of Lemma 36, can be replaced with

2′. the maximal fork rewriting q_j^{f} of q_j is weakly ditree-shaped and there is an $a \in \mathsf{Ind}(\mathcal{A})$ that is not from the core part of $\mathcal{A}$ and satisfies $\mathcal{A}, \mathcal{T}^+ \models A_{q_j^{\mathsf{f}}}(a)$

Consequently, the set $\mathcal{Q}$ used in the construction of $\mathcal{T}^+$ now only needs to contain

- the queries $q_1^{\mathsf{f}}, \ldots, q_\ell^{\mathsf{f}}$;
- the queries $p_1, \ldots, p_k$ from any directed forest decomposition $(p_{\mathsf{core}}, p_1, x_1, \ldots, p_k, x_k, \mu)$ in $\mathsf{fdec}(q_1) \cup \cdots \cup \mathsf{fdec}(q_\ell)$.

It is then a consequence of the following lemma that the size of the extended TBox $\mathcal{T}^+$ is now bounded by $p(|q| + |\mathcal{T}|)$ for some polynomial p. The same arguments as before then allow us to carry over that bound to the number of states in $\mathfrak{A}_Q$. The following is a reformulation of Lemma 4 in [Lutz, 2007]. Note that this result crucially relies on Condition 6 from the definition of forest decompositions.

Lemma 38. *Let q be a connected BCQ and let $\mathcal{Q}$ be the set of all queries that occur as a tree component in a directed forest decomposition of q. Then the cardinality of $\mathcal{Q}$ is polynomial in q.*

We now sketch the construction of the TWAPA $\mathfrak{A}_\mathcal{T}$ from Point 2 of Proposition 13. We first build a TWAPA $\mathfrak{A}$ which accepts a proper $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree (T, L) iff $\mathcal{A}_{(T,L)}$ is inconsistent with $\mathcal{T}$, and then obtain $\mathfrak{A}_\mathcal{T}$ from $\mathfrak{A}$ by complementing and intersecting with $\mathfrak{A}_{Q,1}$.

We can assume that $\mathcal{T}$ is satisfiable because in all considered cases,

1. TBox satisfiability is not harder than the reasoning problem (containment or FO rewritability) that we are interested in;
2. the result of containment or FO rewritability is trivial if at least one of the involved TBoxes is unsatisfiable.

By Lemma 35, we thus have to construct $\mathfrak{A}$ such that it accepts the input tree (T, L) iff a functionality assertion from $\mathcal{T}$ is violated by $\mathcal{A}_{T,L}$ or there is a derivation tree for $\bot(a)$ for some $a \in \mathsf{Ind}(\mathcal{A}_{(T,L)})$. The former is straightforward and the latter can be done in almost exactly the same way as in the automaton $\mathfrak{A}_{Q,2}$ above. To start, we put the following transitions for all $\rho \in \Sigma_\varepsilon$ and all $\nu \in \Sigma_N$, where s_0 is the initial state:

- $\delta(s_0, \rho) = \bigvee_{a\in\mathsf{Ind}(\rho)} \langle 0\rangle s_{\bot,a} \vee \bigvee_{i\in 1..m} \langle i\rangle s_0$;
- $\delta(s_0, \nu) = s_\bot \vee \bigvee_{i\in 1..m} \langle i\rangle s_0$.

It thus remains to deal with the proof obligations $s_{\bot,a}$ and $s_\bot$, which is done as in $\mathfrak{A}_{Q,2}$ except that we use the original TBox $\mathcal{T}$ in place of the extended TBox $\mathcal{T}^+$ (and treat $\bot$ like a concept name from $\mathsf{CN}(\mathcal{T})$). This finishes the construction of the automaton $\mathfrak{A}_\mathcal{T}$. The size of at most $p(|\mathcal{T}|)$ states is easily verified. Note that the construction is essentially independent

of q (except for condition (iv) of properness) because the transitions of $\mathfrak{A}_{Q,2}$ that refer to forest decompositions are replaced by the transitions given above.

C.5 Deciding Containment

We prove the upper bounds for containment stated in Theorem 5. Actually, they follow directly from Proposition 13, the fact that $Q_1 \subseteq Q_2$ (where $Q_i = (\mathcal{T}_i, \Sigma, q_i)$) iff $L(\mathfrak{A}_{Q_1}) \cap L(\mathfrak{A}_{\mathcal{T}_2}) \subseteq L(\mathfrak{A}_{Q_2})$ iff $L(\mathfrak{A}_{Q_1}) \cap L(\mathfrak{A}_{\mathcal{T}_2}) \cap \overline{L(\mathfrak{A}_{Q_2})}$ is empty, the closure of TWAPAs under (polynomial) intersection and complement, and the complexity of TWAPA emptiness.

C.6 Deciding FO rewritability

We prove the upper bounds for FO rewritability stated in Theorems 5 and 6. The proof is based on the characterization from Points 2 (for $\mathcal{ELIHF}_\bot$) and Point 1 (for $\mathcal{ELHF}_\bot$) of Theorem 11 and uses the automata from Proposition 13, in a slightly adapted form.

Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ from ($\mathcal{ELIHF}_\bot$, conBCQ) and $k_0 = 2^{4(|\mathcal{T}|+2^{|q|})^2}$ the bound from Point 2 of Theorem 11. By that theorem, Q is not FO-rewritable iff there is a pseudo tree ABox $\mathcal{A}$ of width at most $|q|$ and outdegree at most $|\mathcal{T}|$ that satisfies the following conditions:

1. $\mathcal{A}$ is consistent with $\mathcal{T}$;
2. $\mathcal{A} \models Q$;
3. $\mathcal{A}|_{>0} \not\models Q$;
4. $\mathcal{A}|_{\leq k_0} \not\models Q$.

We aim to build a TWAPA $\mathfrak{A}$ that accepts representations of such ABoxes; it then remains to decide emptiness. To deal with the 'truncated' ABoxes $\mathcal{A}|_{\leq k_0}$, we need to endow $\Sigma_\varepsilon \cup \Sigma_N$-labeled trees with a counting component. More precisely, we now use $\Sigma_\varepsilon \cup (\Sigma_N \times [k_0])$-labeled trees, where $[k_0] = \{1, \dots, k_0+1\}$. All notions for $\Sigma_\varepsilon \cup \Sigma_N$-labeled trees such as properness, the associated ABox carry over to the extended alphabet. Additionally, we say that a $\Sigma_\varepsilon \cup (\Sigma_N \times [k_0])$-labeled tree (T, L) is *counting* if for every node $x \in T$ on level $i > 0$, $L(T) = (\alpha, j)$ implies $j = \mathsf{min}(i, k_0 + 1)$.

The desired TWAPA $\mathfrak{A}$ is the intersection of five TWAPAs $\mathfrak{A}_0, \dots, \mathfrak{A}_4$. While $\mathfrak{A}_0$ makes sure that the input tree (T, L) is proper and counting, each of the automata $\mathfrak{A}_1, \dots, \mathfrak{A}_4$ makes sure that the ABox $\mathcal{A}_{(T,L)}$ satisfies the corresponding condition from the above list. In fact, we have already seen in Section C.4 how to build TWAPAs for Conditions 1 and 2; they are easily adapted to the new input format and simply ignore the additional counting component of input trees. Moreover, the automaton $\mathfrak{A}_Q$ which ensures Condition 2 is easily modified to ensure Conditions 3 and 4 provided that the input tree is counting; the modified automaton simply ignores those parts of the input that are 'truncated away'.

It thus remains to verify that we can build the automaton $\mathfrak{A}_0$. Properness was already dealt with in Section C.4. We additionally need to verify that the input tree is counting. This can be done with $O(\mathsf{log}(k_0))$ states: we send a copy to the automaton to every tree node, for every i-th bit, $i \in \{1, \dots, \lceil \mathsf{log}(k_0) \rceil\}$. Based on the node label, we determine the value $t \in \{0, 1\}$ of the i-th bit at all successors and send a copy of the automaton in state "bit i=t" to all successors nodes, where that value is verified.

It can be verified that the constructed overall TWAPA $\mathfrak{A}$ has $2^{p(|q_1|+\mathsf{log}(|\mathcal{T}_1|))}$ states, p a polynomial. From the complexity of TWAPA emptiness, we thus get Point 1 in Theorem 6. If $\mathcal{T}$ is formulated in $\mathcal{ELIHF}_\bot$, then $\mathfrak{A}$ has only $p(|q_1| + |\mathcal{T}_1|)$ states due to the improved bounds for this logic in Theorem 11 and Proposition 13. Consequently, we also obtain the upper bound in Point 2 of Theorem 5.

D Rooted Queries

We now establish the coNEXPTIME upper bounds for rooted queries (Theorem 15). We first give the proof for containment, and afterwards we explain how the construction can be modified to handle FO-rewritability.

D.1 Overview of upper bound for containment

For convenience, we repeat the result we aim to prove.

Theorem 39. *Containment for OMQs in ($\mathcal{ELIHF}_\bot$, rCQ) is in co*NEXPTIME.

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be $\mathcal{ELIHF}_\bot$ TBoxes, let Σ be an ABox signature, and let q_1 and q_2 be rooted CQs. We recall that by Proposition 8, $(\mathcal{T}_1, \Sigma, q_1) \not\subseteq (\mathcal{T}_2, \Sigma, q_2)$ iff there is a pseudo tree Σ-ABox $\mathcal{A}$ of outdegree at most $|\mathcal{T}_1|$ and width at most $|q_1|$ that is consistent with both $\mathcal{T}_1$ and $\mathcal{T}_2$ and a tuple $\mathbf{a}$ from the core of $\mathcal{A}$ such that $\mathcal{T}_1, \mathcal{A} \models q_1(\mathbf{a})$ and $\mathcal{T}_2, \mathcal{A} \not\models q_2(\mathbf{a})$. To test for the existence of such a witness ABox and tuple, we proceed as follows.

Step 1 Guess the following:

- pseudo tree Σ-ABox $\mathcal{A}_{\mathsf{init}}$ whose core is bounded by $|q_1|$, whose outdegree is bounded by $|\mathcal{T}_1|$, whose depth is bounded by $m_q = \mathsf{max}(|q_1|, |q_2|)$ (with $\mathcal{U}_q$ the set of individuals that are at distance exactly m_q from the core)
- tuple $\mathbf{a}$ of individuals from the core of $\mathcal{A}$, of the same arity as q_1 and q_2
- ABoxes $\mathcal{B}_1$ and $\mathcal{B}_2$ such that $\mathcal{A} \subseteq \mathcal{B}_i$ and $\mathcal{B}_i \setminus \mathcal{A} \subseteq \{B(a) \mid a \in \mathsf{Ind}(\mathcal{A}), B \in \mathsf{N_C}\}$, for $i \in \{1, 2\}$
- two 'global' candidate transfer sequences $\mathcal{Y}^1_0, \dots, \mathcal{Y}^1_{N_1}$ and $\mathcal{Y}^2_0, \dots, \mathcal{Y}^2_{N_2}$ for $(\mathcal{U}_q, \mathcal{T}_1)$ and $(\mathcal{U}_q, \mathcal{T}_2)$ respectively (precise definition given later)

Intuitively, the guessed ABox $\mathcal{A}_{\mathsf{init}}$ is the initial portion of a witness for non-containment, obtained by restricting the witness ABox to individuals within distance $m_q = \mathsf{max}(|q_1|, |q_2|)$ of the core, and $\mathbf{a}$ is a tuple witnessing the non-containment. The ABox $\mathcal{B}_i$ ($i \in \{1, 2\}$) enriches $\mathcal{A}_{\mathsf{init}}$ with the concept assertions over $\mathsf{Ind}(\mathcal{A})$ that are entailed from the full witness ABox and the TBox $\mathcal{T}_i$. To keep track of the interactions between the guessed part $\mathcal{A}_{\mathsf{init}}$ and the missing trees, we generalize the notion of transfer sequence to sets of individuals (rather than a single individual). The guessed sequence $\mathcal{Y}^i_0, \dots, \mathcal{Y}^i_{N_i}$ ($i \in \{1, 2\}$) corresponds to the transfer sequence of the full witness ABox with respect to the individuals in $\mathcal{U}_q$ (occuring at depth m_q) and the TBox $\mathcal{T}_i$.

Step 2 Verify that:

- for $i \in \{1, 2\}$, $\mathcal{B}_i$ is consistent with $\mathcal{T}_i$

- $\mathcal{B}_1, \mathcal{T}_1 \models q_1(\mathbf{a})$ and $\mathcal{B}_2, \mathcal{T}_2 \not\models q_2(\mathbf{a})$
- for $i \in \{1, 2\}$, the candidate transfer sequence $\mathcal{Y}^i_0, \ldots, \mathcal{Y}^i_{N_i}$ is compatible with $(\mathcal{A}, \mathcal{B}_i)$

and return no if one of these conditions fails to hold.

The second point corresponds to checking that, with respect to the full witness, we have $q_1(\mathbf{a})$ but not $q_2(\mathbf{a})$. Indeed, since q_1 and q_2 are rooted, we know that query matches only involve individuals that are within distance m_q of the core. Since $\mathcal{B}_i$ contains all concept assertions for these individuals that are entailed w.r.t. the full witness ABox, it can be used in place of the witness. The compatibility checks in the third item (which will be made precise further) will be used to ensure that $\mathcal{Y}^i_0, \ldots, \mathcal{Y}^i_{N_i}$ is the transfer sequence of the full witness ABox w.r.t. $\mathcal{U}_q$ and $\mathcal{T}_i$.

Step 3 For each individual $u \in \mathcal{U}_q$, construct a tree automaton that checks whether there is a tree-shaped ABox $\mathcal{A}_u$ rooted at u that does not contain any concept assertion $A(u)$ and is such that for both $i \in \{1, 2\}$, we have:

- $\mathcal{Y}^i_0, \ldots, \mathcal{Y}^i_{N_i}$ is compatible with $\mathcal{A}_u$ at u w.r.t. $\mathcal{T}_i$
- $\mathcal{A}_u \cup \mathcal{Y}^i_{N_i}$ is consistent with $\mathcal{T}_i$
- if $\mathsf{func}(r) \in \mathcal{T}_i$ and $r(u, u') \in \mathcal{A}$, then $\mathcal{A}_u$ does not contain any assertion of the form $r(u, u'')$

Return yes if all of these automata are non-empty, otherwise return no.

The final step checks that it is possible to construct, for every individual u at depth m_q, a tree-shaped ABox $\mathcal{A}_u$, such that the ABox $\mathcal{A}_{\mathsf{init}} \cup \bigcup_{u \in \mathcal{U}_q} \mathcal{A}_u$ that is obtained by attaching all of these trees to $\mathcal{A}_{\mathsf{init}}$ yields the full witness ABox (by renaming individuals, we can assume that $\mathsf{Ind}(\mathcal{A}_{\mathsf{init}}) \cap \mathsf{Ind}(\mathcal{A}_u) = \{u\}$ and $\mathsf{Ind}(\mathcal{A}_u) \cap \mathsf{Ind}(\mathcal{A}_v) = \emptyset$ for $u, v \in$ with $u \neq v$). For this to be the case, we need to ensure that the tree-shaped ABoxes allow us to infer exactly those concept assertions present in the candidate transfer sequence (this is the purpose of the compatibility condition, formalized further). We must also ensure that after adding the entailed assertions $\mathcal{Y}^i_{N_i}$ to ABox $\mathcal{A}_u$, the resulting ABox is consistent with both TBoxes and that no violations of functionality assertions are introduced when attaching $\mathcal{A}_u$ to $\mathcal{A}_{\mathsf{init}}$.

In what follows, we provide more details on Steps 2 and 3 of this procedure.

D.2 Query entailment checks in Step 2

We briefly explain how to perform the query entailment checks in Step 2. We focus on the first entailment check $\mathcal{B}_1, \mathcal{T}_1 \models q_1(\mathbf{a})$, but the same construction can be used to decide whether $\mathcal{B}_2, \mathcal{T}_2 \not\models q_2(\mathbf{a})$. The idea is as follows: to decide whether $\mathcal{B}_1, \mathcal{T}_1 \models q_1(\mathbf{a})$, we will compute the restriction $\mathcal{I}^{q_1}_{\mathcal{B}_1, \mathcal{T}}$ of the canonical model $\mathcal{I}_{\mathcal{B}_1, \mathcal{T}}$ to the ABox individuals in $\mathcal{B}$ and the new domain elements that are within distance $|q_1|$ of one of these individuals. Then it suffices to iterate over all (exponentially many) mappings π from the variables of q_1 into $\Delta^{\mathcal{I}^{q_1}_{\mathcal{B}, \mathcal{T}}}$ and to check if one of these mappings is a match for the query.

We sketch how to construct the interpretation $\mathcal{I}^{q_1}_{\mathcal{B}_1, \mathcal{T}}$ in exponential time. For convenience, we adopt the ABox representation of interpretations. We first include all concept and role assertions that are entailed from $\mathcal{B}_1, \mathcal{T}$; such assertions can be computed in exponential time (e.g., by applying the modified closure rules from Appendix B.2). Next, for each individual $a \in \mathsf{Ind}(\mathcal{B}_1)$, we let C_a be the conjunction of concepts A such that $A(a) \in \mathcal{B}_1$. To determine which successors we need to connect to a, we compute all axioms of the form $C_a \sqsubseteq \exists R.D$, where R is a conjunction of roles from $\mathsf{sig}(\mathcal{T})$ and D is a conjunction of concept names from $\mathsf{sig}(\mathcal{T})$. We keep only the 'strongest' such axioms, i.e. those for which there does not exist an entailed axioms $C_a \sqsubseteq \exists R'.D'$ where R' (resp. D') contains a superset of role (resp. concept) names, and at least one of these superset relationships is strict. It is not hard to show that there can be at most $|\mathcal{T}|$ strongest axioms, one for each existential restriction on the right-hand side of an inclusion in $\mathcal{T}$. If $C_a \sqsubseteq \exists R.D$ is a strongest entailed axiom, and there is no $b \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{T} \models R(a, b)$ and $\mathcal{A}, \mathcal{T} \models D(b)$, then we pick a fresh individual c and add the following assertions: $\{r(a, c) \mid r \in R\} \cup \{A(c) \mid A \in D\}$. For each of the newly introduced individuals, we proceed in exactly the same manner to construct its successors, stopping when an individual has no successors, or when the individual occurs at distance $|q|$ from one of the original individuals. Since the number of successors of an individual is bounded by $|\mathcal{T}|$, and we stop producing successors at depth $|q|$, we only introduce exponentially many individuals. Moreover, to decide which successors to add (and which concepts and roles they should satisfy), we perform at most exponentially many entailment checks, and every such check can be performed in exponential time.

D.3 Transfer sequences for frontier individuals

We next formally introduce the generalized notion of transfer sequence, as well as the candidate transfer sequences that we guess in Step 1.

Consider an arbitrary pseudo tree ABox $\mathcal{A}$. We call a set of individuals $\{u_1, \ldots, u_\ell\} \subseteq \mathsf{Ind}(\mathcal{A})$ a *valid frontier for* $\mathcal{A}$ if there do not exist $u_i \neq u_j$ such that u_i is a descendant of u_j in one of the tree-shaped ABoxes of $\mathcal{A}$. If $\mathcal{U} = \{u_1, \ldots, u_\ell\}$ is a valid frontier for $\mathcal{A}$, then we use $\mathcal{A}^{\uparrow}_{\mathcal{U}}$ to denote the ABox obtained from $\mathcal{A}$ by dropping the subtrees $\mathcal{A}^{\downarrow}_{u_1}, \ldots, \mathcal{A}^{\downarrow}_{u_\ell}$ from $\mathcal{A}$, excepting the individuals $u_1, \ldots, u_\ell$. Slightly abusing notation, we will extend the notation $\mathsf{AT}^{\vdash}_{\mathcal{A}}$ to sets of individuals as follows:

$$\mathsf{AT}^{\vdash}_{\mathcal{A}, \mathcal{T}}(\mathcal{U}) := \{A(u) \mid u \in \mathcal{U}, A(u) \in \mathcal{A}^c_{\mathcal{T}}\}.$$

(Note that we add $\mathcal{T}$ to the subscript to make clear which TBox was used to complete $\mathcal{A}$.)

If $\mathcal{U} = \{u_1, \ldots, u_\ell\}$ is a valid frontier for $\mathcal{A}$, then the *transfer sequence* $\mathcal{X}_0, \mathcal{X}_1, \ldots$ *of* $(\mathcal{A}, \mathcal{U})$ *w. r. t.* $\mathcal{T}$ is defined inductively as follows:

- $\mathcal{X}_0 = \mathsf{AT}^{\vdash}_{\mathcal{A}^0}(\mathcal{U})$, where $\mathcal{A}^0 = \mathcal{A}^{\uparrow}_{\mathcal{U}}$;
- $\mathcal{X}_1 = \mathsf{AT}^{\vdash}_{\mathcal{A}^1}(\mathcal{U})$, where $\mathcal{A}^1 = \bigcup_{u \in \mathcal{U}} \mathcal{A}^{\downarrow}_u \cup \mathcal{X}_0$;
- for $i \geq 0$, $\mathcal{X}_{2i+2} = \mathsf{AT}^{\vdash}_{\mathcal{A}^{2i+2}}(\mathcal{U})$, where $\mathcal{A}^{2i+2} = \mathcal{A}^{2i} \cup \mathcal{X}_{2i+1}$ (equivalently: $\mathcal{A}^{2i+2} = \mathcal{A}^{\uparrow}_{\mathcal{U}} \cup \mathcal{X}_{2i+1}$);

- for $i \geq 1$, $\mathcal{X}_{2i+1} = \mathsf{AT}^{\vdash}_{\mathcal{A}^{2i+1}}(\mathcal{U})$, where $\mathcal{A}^{2i+1} = \mathcal{A}^{2i-1} \cup \mathcal{X}_{2i}$ (equivalently: $\mathcal{A}^{2i+1} = \bigcup_{u \in \mathcal{U}} \mathcal{A}^{\downarrow}_u \cup \mathcal{X}_{2i}$).

An analogue of Lemma 26 can be shown:

Lemma 40. *Let $N = (|\mathcal{U}| \cdot |\mathsf{sig}(\mathcal{T})|) + 1$. Then $\mathcal{X}_N = \mathcal{X}_{N'}$ for all $N' > N$ and $(\mathcal{A}^{N-1})^c_{\mathcal{T}} \cup (\mathcal{A}^N)^c_{\mathcal{T}} = \mathcal{A}^c_{\mathcal{T}}$.*

By *candidate transfer sequence* for $(\mathcal{U}, \mathcal{T})$ we mean a sequence $\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_N$ such that $N = (|\mathcal{U}| \cdot |\mathsf{sig}(\mathcal{T})|) + 1$ and for every $j \geq 0$, $\mathcal{X}_j \subseteq \{A(u_i) \mid A \in \mathsf{N_C} \cap \mathsf{sig}(\mathcal{T}), u_i \in \mathcal{U}\}$ and $\mathcal{X}_j \subseteq \mathcal{X}_{j+1}$. In our procedure, we consider candidate transfer sequences for $(\mathcal{U}_q, \mathcal{T}_1)$ and $(\mathcal{U}_q, \mathcal{T}_2)$, which will terminate by the indices $N_1 = (|\mathcal{U}_q| \cdot |\mathsf{sig}(\mathcal{T}_1)|) + 1$ and $N_2 = (|\mathcal{U}_q| \cdot |\mathsf{sig}(\mathcal{T}_2)|) + 1$, respectively. Observe that $|\mathcal{U}_q| \leq |\mathcal{T}_i|^{m_q}$, so N_i is polynomial in $|\mathcal{T}_i|$ and exponential in $\mathsf{max}(|q_1|, |q_2|)$.

D.4 Compatibility of candidate transfer sequences

Let $\mathcal{A}$ be a pseudo tree ABox and $\mathcal{B} \supseteq \mathcal{A}$ be an ABox with $\mathcal{B} \setminus \mathcal{A} \subseteq \{B(a) \mid a \in \mathsf{Ind}(\mathcal{A}), B \in \mathsf{N_C}\}$. Further let $\mathcal{U} \subseteq \mathsf{Ind}(\mathcal{A})$ be a subset of the leaves of $\mathcal{A}$ (i.e. individuals occurring in one of the trees of $\mathcal{A}$ but without any successors), and let $\mathcal{X} = \mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_N$ be a candidate transfer sequence for $(\mathcal{U}, \mathcal{T})$. We say that $\mathcal{X}$ is *compatible with $(\mathcal{A}, \mathcal{B})$ w.r.t. $(\mathcal{U}, \mathcal{T})$* iff:

1. $\mathcal{X}_0 = \mathsf{AT}^{\vdash}_{\mathcal{A}^0, \mathcal{T}}(\mathcal{U})$, where $\mathcal{D}^0 = \mathcal{A}$;
2. for every $i \geq 0$ with $2i + 2 < N$: $\mathcal{X}_{2i+2} = \mathsf{AT}^{\vdash}_{\mathcal{A}^{2i+2}, \mathcal{T}}(\mathcal{U})$ where $\mathcal{D}^{2i+2} = \mathcal{A} \cup \mathcal{X}_{2i+1}$;
3. $\mathcal{X}_N = \{A(u) \in \mathcal{B} \mid u \in \mathcal{U}\}$;
4. for every $B(a)$ with $a \in \mathsf{Ind}(\mathcal{A})$ and $B \in \mathsf{N_C}$: $B(a) \in \mathcal{B}$ iff $\mathcal{T}, \mathcal{A} \cup \mathcal{X}_N \models B(a)$

Checking compatibility of a candidate transfer sequence w.r.t. a pair of ABoxes can be decided in ExpTime. Indeed, all four conditions involve computing the closure of an exponential-sized ABox w.r.t. $\mathcal{T}$, which can be done in exponential time. Indeed, there are only exponentially many concept assertions that can be added, so only exponentially many rule applications are needed to reach the closure, and finding the next rule to apply involves an exponential number of (ExpTime) entailment checks.

Next let $u \in \mathcal{U}$, and let $\mathcal{G}$ be a tree-shaped ABox with root u. We say that $\mathcal{X}$ is *compatible with $\mathcal{G}$ at u w.r.t. $\mathcal{T}$* if and only if:

- $\mathcal{X}_1(u) = \mathsf{AT}^{\vdash}_{\mathcal{D}^1, \mathcal{T}}(u)$ where $\mathcal{D}^1 = \mathcal{G} \cup \mathcal{X}_0(u)$
- for every $i \geq 1$ with $2i + 1 \leq N$: $\mathcal{X}_{2i+1}(u) = \mathsf{AT}^{\vdash}_{\mathcal{D}^{2i+1}, \mathcal{T}}(u)$, where $\mathcal{D}^{2i+1} = \mathcal{G} \cup \mathcal{X}_{2i}(u)$

where, slightly abusing notation, we use the notation $\mathcal{X}_i(u)$ to mean the set $\{A(u) \mid A(u) \in \mathcal{X}_i\}$.

D.5 Automata construction

Let $\mathcal{A}$ be the guessed pseudo tree ABox, let $\mathcal{Y}^1$ and $\mathcal{Y}^2$ be the guessed candidate transfer sequences for $(\mathcal{U}_q, \mathcal{T}_1)$ and $(\mathcal{U}_q, \mathcal{T}_2)$ respectively, and let $\mathcal{U}_q = \{u_1, \ldots, u_{\ell_q}\}$.

To implement Step 3 of the procedure, we need to construct, for every $1 \leq j \leq \ell_q$, a TWAPA $\mathfrak{A}_j$ that accepts encodings of tree-shaped ABoxes $\mathcal{G}_j$ with root node u_j satisfying the conditions of Proposition 41. The desired automaton $\mathfrak{A}_j$ can be obtained by intersecting the following automaton:

- $\mathfrak{A}_{\mathsf{cons}}$ that ensures that the encoded ABox is consistent with the TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$.
- $\mathfrak{A}_{\mathsf{funct}}$ that ensures that the encoded ABox, when added to the ABox $\mathcal{A}$, does not violate the functionality assertions in $\mathcal{T}_1$ and $\mathcal{T}_2$. Specifically, we need to ensure that if $\mathsf{func}(r) \in \mathcal{T}_k$ ($k \in \{1, 2\}$) and $r(u_j, u') \in \mathcal{A}$, then the encoded ABox (whose root individual is u_j) does not contain any assertion of the form $r(u_j, u'')$.
- for every $1 \leq 2i + 1 \leq N$ and $k \in \{1, 2\}$, an automaton $\mathfrak{A}^{\mathcal{Y}^k}_{2i+1}$ that determines whether $\mathcal{Y}^k_{2i+1}(u_j)$ is precisely the set of concept assertions about u_j that are entailed from $\mathcal{T}$, the encoded ABox, and the assertions in $\mathcal{Y}^k_{2i}(u_j)$.

Note that the automaton $\mathfrak{A}^{\mathcal{Y}^k}_{2i+1}$ in the third item can be constructed by intersecting automata that check whether a given concept assertion $A(u_j) \in \mathcal{Y}^k_{2i+1}(u_j)$ is entailed with those checking that each concept assertion $A(u_j) \notin \mathcal{Y}^k_{2i+1}(u_j)$ (with $A \in \mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})$) is not entailed. Moreover, the automata checking whether a concept is not entailed at the root u_j can be obtained by complementing the automaton that accepts trees in which the concept is entailed at the root.

Importantly, because the sets $\mathcal{Y}_i(u_j)$ increase monotonically and only contain concept assertions about the individual u_j, there are only *polynomially many* different elements in the set $\{(\mathcal{Y}_{2i+1}(u_j), \mathcal{Y}_{2i}(u_j)) \mid 1 \leq 2i + 1 \leq N\}$. It follows that $\mathfrak{A}_j$ can be obtained by *intersecting a polynomial number of automata*. Moreover, it is not hard to see that each of the component automata can be *constructed in polynomial time*. Since there are (at most) single exponentially many elements in $\mathcal{U}_q$, and emptiness of TWAPAs can be tested in single-exponential time, it follows that Step 3 can be performed in ExpTime.

D.6 Correctness of the procedure

We have already given the main lines of the argument in the overview, so here we concentrate on the following proposition, which is the key step to establishing correctness.

Proposition 41. *Let $\mathcal{A}$ be a pseudo tree ABox, let $\mathcal{B} \supseteq \mathcal{A}$ be an ABox with $\mathcal{B} \setminus \mathcal{A} \subseteq \{B(a) \mid a \in \mathsf{Ind}(\mathcal{A}), B \in \mathsf{N_C}\}$ that is consistent with $\mathcal{T}$, and let $\mathcal{U} = \{u_1, \ldots, u_\ell\} \subseteq \mathsf{Ind}(\mathcal{A})$ be a subset of the leaves in $\mathcal{A}$. Suppose that*

1. *the candidate transfer sequence $\mathcal{X} = \mathcal{X}_0, \ldots, \mathcal{X}_N$ is compatible with $(\mathcal{A}, \mathcal{B})$ w.r.t. $(\mathcal{U}, \mathcal{T})$, and*
2. *there exist tree-shaped ABoxes $\mathcal{G}_1, \ldots, \mathcal{G}_\ell$ such that $\mathsf{Ind}(\mathcal{G}_j) \cap \mathsf{Ind}(\mathcal{G}'_j) = \emptyset$ for every $j \neq j'$ and for every $1 \leq j \leq \ell$:*
   - *$\mathsf{Ind}(\mathcal{A}) \cap \mathsf{Ind}(\mathcal{G}_j) = \{u_j\}$,*
   - *$\mathcal{G}_j$ does not contain any concept assertion $A(u_j)$,*
   - *$\mathcal{G}_j \cup \mathcal{X}_N$ is consistent with $\mathcal{T}$,*
   - *$\mathcal{A} \cup \mathcal{G}_j$ does not violate any functionality assertion in $\mathcal{T}$,*
   - *$\mathcal{G}_j$ is compatible with $\mathcal{X}$ at u_j w.r.t. $\mathcal{T}$.*

Let $\mathcal{A}^* = \mathcal{A} \cup \bigcup_{1 \leq j \leq \ell} \mathcal{G}_j$. Then:

- *$\mathcal{A}^*$ is consistent with $\mathcal{T}$,*
- *$\mathcal{X}$ is the transfer sequence for $(\mathcal{A}^*, \{u_1, \ldots, u_\ell\})$, and*

- $\mathcal{T}, \mathcal{A}^* \models A(a)$ *iff* $A(a) \in \mathcal{B}$ *(for* $a \in \mathsf{Ind}(\mathcal{A})$*,* $A \in \mathsf{N_C}$*)*

Proof. Let $\mathcal{A}, \mathcal{B}, \mathcal{G}_1, \ldots, \mathcal{G}_\ell, \mathcal{U}, \mathcal{T}, \mathcal{X}$ be as in the statement.

To show consistency of $\mathcal{A}^*$ with $\mathcal{T}$, first note that $\mathcal{A}^*$ does not violate any functionality assertions in $\mathcal{T}$, since each of the ABoxes $\mathcal{A}, \mathcal{G}_1, \ldots, \mathcal{G}_\ell$ is consistent with $\mathcal{T}$, there is no individual shared by two different $\mathcal{G}_j$, and by assumption, for every $1 \leq j \leq \ell$, the ABox $\mathcal{A} \cup \mathcal{G}_j$ does not violate any functionality assertion in $\mathcal{T}$.

Let $\mathcal{A}^+ = \mathcal{A} \cup \mathcal{X}_N$, and let $\mathcal{G}_j^+ = \mathcal{G}_j \cup \mathcal{X}_N(u_j)$ for $1 \leq j \leq \ell$. We know that each of the ABoxes $\mathcal{A}^+, \mathcal{G}_1^+, \ldots, \mathcal{G}_\ell^+$ is consistent with $\mathcal{T}$, and hence possesses a canonical model. We let $\mathcal{I}_{\mathcal{A}^+,\mathcal{T}}$ and $\mathcal{I}_{\mathcal{G}_j^+,\mathcal{T}}$ be the canonical models for $\mathcal{A}+, \mathcal{T}$ (resp. $\mathcal{G}_j^+, \mathcal{T}$), as defined in Appendix B.2. Without loss of generality, we may assume that $\Delta^{\mathcal{I}_{\mathcal{G}_j,\mathcal{T}}} \cap \Delta^{\mathcal{I}_{\mathcal{G}_k,\mathcal{T}}} = \emptyset$ for every $j \neq k$, and that $\Delta^{\mathcal{I}_{\mathcal{A},\mathcal{T}}} \cap \Delta^{\mathcal{I}_{\mathcal{G}_j,\mathcal{T}}} = \emptyset$ for every $1 \leq j \leq \ell$. We recall that these interpretations can be seen as adding to the original ABox all entailed assertions about the ABox individuals, and additionally attaching weakly tree-shaped interpretations to each of the ABox individuals in order to witness the existential restrictions on the right-hand side of TBox axioms. For every $u_j \in \mathcal{U}$, we let $\mathcal{I}_j^{\mathcal{A}}$ (resp. $\mathcal{I}_j^{\mathcal{G}}$) be the weakly tree-shaped interpretation that is attached to the individual u_j in $\mathcal{I}_{\mathcal{A},\mathcal{T}}$ (resp. $\mathcal{I}_{\mathcal{G}_k,\mathcal{T}}$). Define the interpretation $\mathcal{J}$ as the union of the interpretations $\mathcal{A}^+, \mathcal{G}_1^+, \ldots, \mathcal{G}_\ell^+$:

- $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}_{\mathcal{A},\mathcal{T}}} \cup \bigcup_{1 \leq j \leq \ell} \Delta^{\mathcal{I}_{\mathcal{G}_j,\mathcal{T}}}$
- for every $A \in \mathsf{N_C}$: $A^{\mathcal{J}} = A^{\mathcal{I}_{\mathcal{A},\mathcal{T}}} \cup \bigcup_{1 \leq j \leq \ell} A^{\mathcal{I}_{\mathcal{G}_j,\mathcal{T}}}$
- for every $r \in \mathsf{N_R}$: $r^{\mathcal{J}} = r^{\mathcal{I}_{\mathcal{A},\mathcal{T}}} \cup \bigcup_{1 \leq j \leq \ell} r^{\mathcal{I}_{\mathcal{G}_j,\mathcal{T}}}$

To satisfy the functionality assertions, we proceed as follows. For every $u_j \in \mathcal{U}$ and functional role R:

- If $u_j \in \exists R^{\mathcal{I}_{\mathcal{A}^+,\mathcal{T}}}$ and there is no b such that $\mathcal{A}^+, \mathcal{T} \models R(u_j, b)$, then let e be the unique element in $\Delta^{\mathcal{I}_{\mathcal{A},\mathcal{T}}}$ such that $(u_j, e) \in R^{\mathcal{I}_{\mathcal{A},\mathcal{T}}}$. This element must belong to the weakly tree-shaped subinterpretation $\mathcal{I}_j^{\mathcal{A}}$. Remove the element e and all of its descendants in $\mathcal{I}_j^{\mathcal{A}}$ from $\Delta^{\mathcal{J}}$.
- If $u_j \in \exists R^{\mathcal{I}_{\mathcal{G}_j^+,\mathcal{T}}}$ and there is no b such that $\mathcal{G}_j^+, \mathcal{T} \models R(u_j, b)$, then let e be the unique element in $\Delta^{\mathcal{I}_{\mathcal{A},\mathcal{T}}}$ such that $(u_j, e) \in R^{\mathcal{I}_{\mathcal{G}_j^+,\mathcal{T}}}$. This element must belong to the weakly tree-shaped subinterpretation $\mathcal{I}_j^{\mathcal{G}}$. Remove the element e and all of its descendants in $\mathcal{I}_j^{\mathcal{G}}$ from $\Delta^{\mathcal{J}}$.

Call the resulting interpretation $\mathcal{J}^-$. We claim that $\mathcal{J}^-$ is a model of $\mathcal{A}^*$ and $\mathcal{T}$. First observe that $\mathcal{J}^-$ makes true all ABox assertions in $\mathcal{A}^*$, since $\mathcal{J}$ satisfies this property and the modifications that were made to $\mathcal{J}$ only involve elements that did not occur in the original ABoxes. Because of our modifications, we have resolved all of the violations of functionality axioms that were introduced when combining the interpretations. It can also be easily seen that axioms of the forms $A \sqsubseteq \bot$, $\top \sqsubseteq A$, $B_1 \sqcap B_2 \sqsubseteq A$, and $\exists r.B \sqsubseteq A$ are all satisfied in $\mathcal{J}^-$, since they were satisfied in each of the interpretations $\mathcal{I}_{\mathcal{A}^+,\mathcal{T}}, \mathcal{I}_{\mathcal{G}_j^+,\mathcal{T}}, \ldots, \mathcal{I}_{\mathcal{G}_j^+,\mathcal{T}}$. Finally, if $e \in A^{\mathcal{J}^-}$ and $A \sqsubseteq \exists r.B \in \mathcal{T}$, then either we have the same witnessing r-successor e' as was used in the component interpretation containing the element e, or $e \in \mathcal{U}$, and we were only allowed to remove e' (and the whole tree-shaped interpretation rooted at e') because in the ABox $\mathcal{A}^*$, there was an ABox individual that acted as the witnessing r-successor. (and which is present in $\mathcal{J}^-$) Thus, $\mathcal{J}^-$ is a model of $\mathcal{A}^*, \mathcal{T}$, so $\mathcal{A}^*$ is consistent with $\mathcal{T}$.

We next prove by induction that $\mathcal{X} = \mathcal{X}_0, \ldots, \mathcal{X}_n$ is the transfer sequence for $(\mathcal{A}^*, \mathcal{U})$. We start by considering the first set in the sequence ($\mathcal{X}_0$). We know that $\mathcal{X}_0 = \mathsf{AT}^{\vdash}_{\mathcal{A},\mathcal{T}}(\mathcal{U})$ since $\mathcal{X}$ is compatible with $(\mathcal{A}, \mathcal{B})$ w.r.t. $(\mathcal{U}, \mathcal{T})$. We then use the fact that, for every $1 \leq j \leq \ell$, the ABox $\mathcal{G}_j$ is such that $\mathsf{Ind}(\mathcal{A}) \cap \mathsf{Ind}(\mathcal{G}_j) = \{u_j\}$ and does not contain any assertion $A(u_j)$ to infer that $(\mathcal{A}^*)^{\uparrow}_{\mathcal{U}} = \mathcal{A}$. Thus, $\mathcal{X}_0$ is the first element in the transfer sequence for $(\mathcal{A}^*, \mathcal{U})$.

For the second element $\mathcal{X}_1$, we first note that, for every $0 \leq i \leq n$, $\mathcal{X}_i = \bigcup_{1 \leq j \leq \ell} \mathcal{X}_i(u_j)$. Further note that for every $1 \leq j \leq \ell$, by the compatibility of $\mathcal{X}$ with $\mathcal{G}_j$ at u_j, we have $\mathcal{X}_1(u_j) = \mathsf{AT}^{\vdash}_{\mathcal{D}_j^1,\mathcal{T}}(u_j)$ where $\mathcal{D}_j^1 = \mathcal{G}_j \cup \mathcal{X}_0(u_j)$. Let $\mathcal{D}^1 = \bigcup_{1 \leq j \leq \ell} \mathcal{D}_j^1$. Since $\mathsf{Ind}(\mathcal{D}_j^1) = \mathsf{Ind}(\mathcal{G}_j)$ and we know that $\mathsf{Ind}(\mathcal{G}_j) \cap \mathsf{Ind}(\mathcal{G}'_j) = \emptyset$ for every $j \neq j'$, it follows that $\mathsf{AT}^{\vdash}_{\mathcal{D}_j^1,\mathcal{T}}(u_j) = \mathsf{AT}^{\vdash}_{\mathcal{D}^1,\mathcal{T}}(u_j)$, and hence that $\mathcal{X}_1 = \mathsf{AT}^{\vdash}_{\mathcal{D}^1,\mathcal{T}}(\mathcal{U})$. Finally, we note that since $(\mathcal{A}^*)^{\downarrow}_{u_j} = \mathcal{G}_j$, we have that $(\mathcal{A}^*)^{\downarrow}_{\mathcal{U}} = \bigcup_{1 \leq j \leq \ell} \mathcal{G}_j$, and thus $\mathcal{D}^1 = (\mathcal{A}^*)^{\downarrow}_{\mathcal{U}} \cup \mathcal{X}_0$, which shows that $\mathcal{X}_1$ is as desired.

Next consider an index $1 < 2i + 2 \leq N$. As we know that $\mathcal{X}$ is compatible with $(\mathcal{A}, \mathcal{B})$ w.r.t. $(\mathcal{U}, \mathcal{T})$, we can infer that $\mathcal{X}_{2i+2} = \mathsf{AT}^{\vdash}_{\mathcal{D}^{2i+2},\mathcal{T}}(\mathcal{U})$ where $\mathcal{D}^{2i+2} = \mathcal{A} \cup \mathcal{X}_{2i+1}$. Since $(\mathcal{A}^*)^{\uparrow}_{\mathcal{U}} = \mathcal{A}$, it follows that $\mathcal{X}_{2i+2}$ is the correct $2i + 2$th element in the transfer sequence for $(\mathcal{A}^*, \mathcal{U})$.

Finally consider an index $0 < 2i + 1 \leq N$. We know that for every $1 \leq j \leq \ell$, the ABox $\mathcal{G}_j$ is compatible with $\mathcal{X}$ at u_j w.r.t. $\mathcal{T}$, so we have $\mathcal{X}_{2i+1}(u) = \mathsf{AT}^{\vdash}_{\mathcal{D}_j^{2i+1},\mathcal{T}}(u)$, where $\mathcal{D}_j^{2i+1} = \mathcal{G}_j \cup \mathcal{X}_{2i}(u)$. Let $\mathcal{D}^{2i+1} = \bigcup_{1 \leq j \leq \ell} \mathcal{D}_j^{2i+1}$. Using the same arguments as for $\mathcal{X}_1$, we can show that $\mathcal{X}_{2i+1} = \mathsf{AT}^{\vdash}_{\mathcal{D}^{2i+1},\mathcal{T}}(\mathcal{U})$ and that $\mathcal{D}^{2i+1} = (\mathcal{A}^*)^{\downarrow}_{\mathcal{U}} \cup \mathcal{X}_{2i}$, as required.

Now we show the third point. For the right-to-left direction, we note that since $\mathcal{X}$ is compatible with $(\mathcal{A}, \mathcal{B})$ w.r.t. $(\mathcal{U}, \mathcal{T})$, we have $\mathcal{T}, \mathcal{A} \cup \mathcal{X}_N \models B(a)$ for every $B(a) \in \mathcal{B}$. Since $\mathcal{X}$ is the transfer sequence for $\mathcal{A}^*$ w.r.t. $\mathcal{T}$, we must have $\mathcal{A}^*, \mathcal{T} \models \mathcal{X}_N$, and by definition, $\mathcal{A}^*$ contains $\mathcal{A}$. It follows that $\mathcal{T}, \mathcal{A}^* \models B(a)$ for every $B(a) \in \mathcal{B}$.

For the left-to-right direction, suppose that $\mathcal{T}, \mathcal{A}^* \models B(a)$, where $a \in \mathsf{Ind}(\mathcal{A})$ and $B \in \mathsf{N_C}$. Then $B(a) \in (\mathcal{A}^*)^c_{\mathcal{T}}$. As $\mathcal{X}$ is the transfer sequence for $(\mathcal{A}^*, \mathcal{U})$, it follows from Lemma 40 that we have $(\mathcal{A}^*)^c_{\mathcal{T}} = (\mathcal{D}_{N-1})^c_{\mathcal{T}} \cup (\mathcal{D}_N)^c_{\mathcal{T}}$ where $\mathcal{D}_{N-1} = (\mathcal{A}^*)^{\downarrow}_{\mathcal{U}} \cup \mathcal{X}_{N-1}$ and $\mathcal{D}_N = (\mathcal{A}^*)^{\uparrow}_{\mathcal{U}} \cup \mathcal{X}_N$. First suppose that $B(a) \in (\mathcal{D}_N)^c_{\mathcal{T}}$. Since $(\mathcal{A}^*)^{\uparrow}_{\mathcal{U}} = \mathcal{A}$, we have $\mathcal{D}_N = \mathcal{A} \cup \mathcal{X}_{N-1}$, so $\mathcal{T}, \mathcal{A} \cup \mathcal{X}_{N-1} \models B(a)$. As $\mathcal{X}_{N-1} \subseteq \mathcal{X}_N$, we also have $\mathcal{T}, \mathcal{A} \cup \mathcal{X}_{N-1} \models B(a)$, which implies $B(a) \in \mathcal{B}$, due to the fourth condition of compatibility of $\mathcal{X}$ with $(\mathcal{A}, \mathcal{B})$ w.r.t. $(\mathcal{U}, \mathcal{T})$. Now consider the case in which $B(a) \in (\mathcal{D}_{N-1})^c_{\mathcal{T}}$, but $B(a) \notin (\mathcal{D}_N)^c_{\mathcal{T}}$. Since $\mathcal{D}_{N-1} = (\mathcal{A}^*)^{\downarrow}_{\mathcal{U}} \cup \mathcal{X}_{N-2}$, we

must have $a \in \mathcal{U}$, and from $B(a) \in (\mathcal{D}_{N-1})^c_{\mathcal{T}}$, we obtain $B(a) \in \mathcal{X}_{N-1}$. It follows that $B(a) \in \mathcal{D}^N$, which contradicts our assumption that $B(a) \notin (\mathcal{D}_N)^c_{\mathcal{T}}$. □

D.7 Upper bound for FO-rewritability

We aim to prove the following:

Theorem 42. *FO-rewritability in ($\mathcal{ELIHF}_\bot$, rCQ) is in co*NEXPTIME.

By Lemma 28, we know that $(\mathcal{T}, \Sigma, q(\mathbf{x}))$ is *not* FO-rewritable iff there exists a k_0-entailment witness for $\mathcal{T}$, Σ, and $q(\mathbf{x})$ of outdegree bounded by $|\mathcal{T}|$ for $k_0 = |q| + 2^{3m^2}$ where $m = |\mathcal{T}|$. Thus, it suffices to provide an NEXPTIME procedure for deciding whether such a witness exists.

The procedure will be quite similar to the NEXPTIME procedure for testing non-containment of rooted queries. In what follows, we outline the main differences.

In Step 1, the guessed ABox $\mathcal{A}_{\mathsf{init}}$ corresponds to initial portion of the k_0-entailment witness (up to depth $|q|$), the tuple $\mathbf{a}$ is the answer tuple associated with the witness, and we take $\mathcal{U}_q$ to be the set of individuals that occur in $\mathcal{A}_{\mathsf{init}}$ at depth $|q|$. In place of the ABoxes $\mathcal{B}_1$ and $\mathcal{B}_2$, we guess two ABoxes $\mathcal{B}$ and $\mathcal{B}_{k_0}$, with the former being used for the concept assertions involving the individuals in $\mathcal{A}_{\mathsf{init}}$ that hold in the full entailment witness (i.e. once we have added back the missing trees), and the latter containing only those assertions that can be obtained using the entailment witness cut off at depth k_0. We also guess two candidate transfer sequences $\mathcal{Y} = \mathcal{Y}_0, \dots, \mathcal{Y}_N$ and $\mathcal{Z} = \mathcal{Z}_0, \dots, \mathcal{Z}_N$ (with $N = (|\mathcal{U}| \cdot |\mathsf{sig}(\mathcal{T})|) + 1$), both with respect to $(\mathcal{U}_q, \mathcal{T})$. The first sequence $\mathcal{Y}$ is intended to track concept entailments w.r.t. the full entailment witness, and the second is for the ABox obtained by restricting the entailment witness to those individuals that occur at depth k_0 or less.

In Step 2, we test whether $\mathcal{T}, \mathcal{B} \models q(\mathbf{a})$ and $\mathcal{T}, \mathcal{B}_{k_0} \not\models q(\mathbf{a})$. We also verify that the first candidate transfer sequence $\mathcal{Y}$ is compatible with $(\mathcal{A}_{\mathsf{init}}, \mathcal{B})$ and the second candidate transfer sequence $\mathcal{Z}$ is compatible with $(\mathcal{A}_{\mathsf{init}}, \mathcal{B}_{k_0})$.

In Step 3, for each $u_j \in U_q$, we build an automaton $\mathfrak{A}_j$ that accepts (encodings of) pseudo tree ABoxes $\mathcal{G}_j$ such that:

- $\mathsf{Ind}(\mathcal{A}_{\mathsf{init}}) \cap \mathsf{Ind}(\mathcal{G}_j) = \{u_j\}$,
- $\mathcal{G}_j$ is consistent with $\mathcal{T}$,
- $\mathcal{A}_{\mathsf{init}} \cup \mathcal{G}_j$ does not violate any functionality assertion in $\mathcal{T}$,
- $\mathcal{G}_j$ is compatible with $\mathcal{Y}$ at u_j w.r.t. $\mathcal{T}$,
- $\mathcal{G}_j|_{\leq k_0 - |q|}$ is compatible with $\mathcal{Z}$ at u_j w.r.t. $\mathcal{T}$.

Note that in the last item, we cut off $\mathcal{G}_j$ at depth $k_0 - |q|$ so that when we attach it to $\mathcal{A}_{\mathsf{init}}$ (in which u_j occurs at depth $|q|$), we obtain an ABox having depth k_0.

Using similar arguments as for containment, we can show that the modified procedure runs in NEXPTIME and it returns yes just in the case that $(\mathcal{T}, \Sigma, q(\mathbf{x}))$ is not FO-rewritable.

E Lower bounds

E.1 coNExpTime lower bounds for rooted CQs

An *(exponential torus) tiling problem* P is a triple (T, H, V), where $T = \{0, \dots, k\}$ is a finite set of *tile types* and $H, V \subseteq T \times T$ represent the *horizontal and vertical matching conditions*. An *initial condition* for P takes the form $c = (c_0, \dots, c_{n-1}) \in T^n$. A mapping $\tau : \{0, \dots, 2^n - 1\} \times \{0, \dots, 2^n - 1\} \to T$ is a *solution* for P given c if for all $x, y < 2^n$, the following holds (where $\oplus_i$ denotes addition modulo i):

- if $\tau(x, y) = t_1$ and $\tau(x \oplus_{2^n} 1, y) = t_2$, then $(t_1, t_2) \in H$
- if $\tau(x, y) = t_1$ and $\tau(x, y \oplus_{2^n} 1) = t_2$, then $(t_1, t_2) \in V$
- $\tau(i, 0) = c_i$ for all $i < n$.

It is well-known that there exists a tiling problem $P = (T, H, V)$ such that, given an initial condition c, it is NEXPTIME-complete to decide whether there exists a solution for P given c. For the following constructions, we fix such a P.

Lemma 43. *Given an input c for P of length n, one can construct in polynomial time an $\mathcal{ELI}$ TBox $\mathcal{T}_c$, a rooted CQ $q_c(x)$, and an ABox signature Σ_c such that, for a selected concept name $A^* \notin \Sigma_c$,*

1. *P has a solution given c iff there is a Σ_c-ABox $\mathcal{A}$ and an $a \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{T}_c \models A^*(a)$ and $\mathcal{A}, \mathcal{T}_c \not\models q_c(a)$;*
2. *there is an $\mathcal{ELI}$-concept C_{q_c} such that $d \in C^{\mathcal{I}}_{q_c}$ implies $\mathcal{I} \models q_c(d)$ for all interpretations $\mathcal{I}$ and $d \in \Delta^{\mathcal{I}}$;*
3. *q_c is FO-rewritable relative to $\mathcal{T}_c$ and Σ_c.*

We will now prove the containment and FO-rewritability lower bounds, assuming the previous lemma. The proof of the lemma is given in the following subsection.

Theorem 44. *Containment in $(\mathcal{ELI}, rCQ)$ is* coNEXPTIME*-hard.*

Proof. Let c be an input to P, and let $\mathcal{T}_c$, $q_c(x)$, Σ_c, and A^* be as in Lemma 43. By Condition 1 of Lemma 43, $(\mathcal{T}_c, \Sigma_c, A^*) \not\subseteq (\mathcal{T}_c, \Sigma_c, q_c)$ over Σ_c-ABoxes iff P has a solution given c. □

Theorem 45. *FO-rewritability in $(\mathcal{ELI}, rCQ)$ is* coNEXPTIME*-hard.*

Proof. Let c be an input to P, and let $\mathcal{T}_c$, $q_c(x)$, Σ_c, and A^* be as in Lemma 43. We obtain a TBox $\mathcal{T}$ by extending $\mathcal{T}_c$ with the following:

$$\begin{aligned} \exists r.A &\sqsubseteq A \\ A \sqcap A^* &\sqsubseteq C_{q_c} \end{aligned}$$

where A and r do not occur in $\mathcal{T}_c$ and q_c, $A^* \notin \Sigma_c$ is the concept name from Lemma 43 and C_{q_c} the concept from Point 2 of that lemma. Set $\Sigma = \Sigma_c \cup \{A, r\}$. It remains to prove the following.

Claim. P has a solution given c iff q_c is not FO-rewritable relative to $\mathcal{T}$ and Σ.

First assume that P has a solution given c. By Point 1 of Lemma 43, there is a Σ_c-ABox $\mathcal{A}$ and an $a_0 \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{T}_c \models A^*(a_0)$ and $\mathcal{A}, \mathcal{T}_c \not\models q_c(a_0)$. Since every $\mathcal{ELI}$ TBox is unraveling tolerant [Lutz and Wolter, 2012] and by compactness, we can assume w.l.o.g. that $\mathcal{A}$ is tree-shaped

with root a_0. Let ℓ be the depth of $\mathcal{A}$. For each $k > \ell$, let $\mathcal{A}_k$ be the ABox obtained by extending $\mathcal{A}$ with

$$r(a_0, a_1), \ldots, r(a_{k-1}, a_k), A(a_k)$$

where $a_k, \ldots, a_1$ do not occur in $\mathcal{A}$. Note that $\mathcal{A}_k$ is tree-shaped and of depth at least k. Since $\mathcal{A}, \mathcal{T}_c \models A^*(a_0)$, it follows from Point 2 of Lemma 43 that $\mathcal{A}_k, \mathcal{T} \models q_c(a_0)$. Now consider the ABox $\mathcal{A}_k|_{\leq k-1}$. We aim to show that $\mathcal{A}_k|_{\leq k-1}, \mathcal{T} \not\models q_c(a_0)$ and then to apply Theorem 9 to show that q_c is not FO-rewritable relative to $\mathcal{T}$ and Σ. Note that $\mathcal{A}_k|_{\leq k-1}$ does not contain A. On such ABoxes, $\mathcal{T}$ can be replaced with $\mathcal{T}_c$ since the left-hand sides of the concept inclusions in $\mathcal{T}$ will never apply. It thus suffices to show that $\mathcal{A}_k|_{k-1}, \mathcal{T}_c \not\models q_c(a_0)$. This follows from $\mathcal{A}, \mathcal{T}_c \not\models q_c(a_0)$ and the fact that r (the only symbol in assertions from $(\mathcal{A}_k|_{k-1}) \setminus \mathcal{A}$) occurs neither in $\mathcal{T}_c$ nor in q_c.

Now assume that P has no solution given c. Let $\widehat{q}_c(x)$ be an FO-rewriting of $q_c(x)$ relative to $\mathcal{T}_c$ and Σ_c. We argue that $\widehat{q}_c(x)$ is also an FO-rewriting of $q_c(x)$ relative to $\mathcal{T}$ and Σ.

First assume $\mathcal{A} \models \widehat{q}_c(a)$ for some Σ-ABox $\mathcal{A}$. Since $\widehat{q}_c(x)$ uses only symbols from Σ_c, this means that $\mathcal{A}' \models \widehat{q}_c(a)$ where $\mathcal{A}'$ is the reduct of $\mathcal{A}$ to symbols in Σ_c. Thus $\mathcal{A}', \mathcal{T}_c \models q_c(a)$, implying $\mathcal{A}, \mathcal{T} \models q_c(a)$.

Conversely, assume that $\mathcal{A}, \mathcal{T} \models q_c(a)$ for some Σ-ABox $\mathcal{A}$ and $a \in \mathsf{Ind}(\mathcal{A})$. Using canonical models and the construction of $\mathcal{T}$, one can show that this implies (i) $\mathcal{A}, \mathcal{T}_c \models q_c(a)$ or (ii) $\mathcal{A}, \mathcal{T}_c \models A^*(a)$. In Case (i), we get $\mathcal{A}', \mathcal{T}_c \models q_c(a)$, where $\mathcal{A}'$ is the Σ_c-reduct of $\mathcal{A}$. Thus $\mathcal{A}' \models \widehat{q}_c(a)$, which implies $\mathcal{A} \models \widehat{q}_c(a)$. In Case (ii), Point 1 of Lemma 43 yields $\mathcal{A}, \mathcal{T}_c \models q_c(a)$ and thus we can proceed as in Case (i). □

E.2 Proof of Lemma 43

Let $c = (c_0, \ldots, c_{n-1})$ be an input for P. We show how to construct the TBox $\mathcal{T}_c$, query q_c, and ABox signature Σ_c that satisfy Points 1 and 2 of Lemma 43. We will first use a UCQ for q_c and later show how to improve to a CQ. The general idea is that $\mathcal{T}_c$ verifies in a bottom-up way the existence of (a homomorphic image of) what we call a *torus tree* in the ABox. A torus tree represents the $2^n \times 2^n$-torus along with a tiling that respects the tiling conditions in P and initial condition c, except that the representation might be defective in that there can be different elements which represent the same grid node but are labeled with different tile types. If a torus tree is found, then $\mathcal{T}_c$ ensures that A^* is derived at the root of the tree. The query q_c will be constructed to become true at the root if and only if the torus tree has a defect. It can then be verified that

(∗) there is a solution for P given c if and only if there is an ABox $\mathcal{A}$ and an $a \in \mathsf{Ind}(\mathcal{A})$ with $\mathcal{A}, \mathcal{T}_c \models A^*(a)$ and $\mathcal{A}, \mathcal{T}_c \not\models q_c(a)$

where intuitively $\mathcal{A}$ is a defect-free torus tree with root a.

Torus trees are of depth $2n+2$ and all tree edges are labeled with the role composition $r^-; r$, where r is the only role name used in the reduction. For readability, we use S to abbreviate $r^-; r$. For example, $\exists S.C$ stands for $\exists r^-.\exists r.C$. Note that S behaves like a reflexive-symmetric role. The ABox signature Σ_c consists of the following symbols:

1. concept names $A_0, \ldots, A_{2n-1}$ and $\overline{A}_0, \ldots, \overline{A}_{2n-1}$ that serve as bits in the binary representation of a number between 0 and $2^{2n} - 1$;

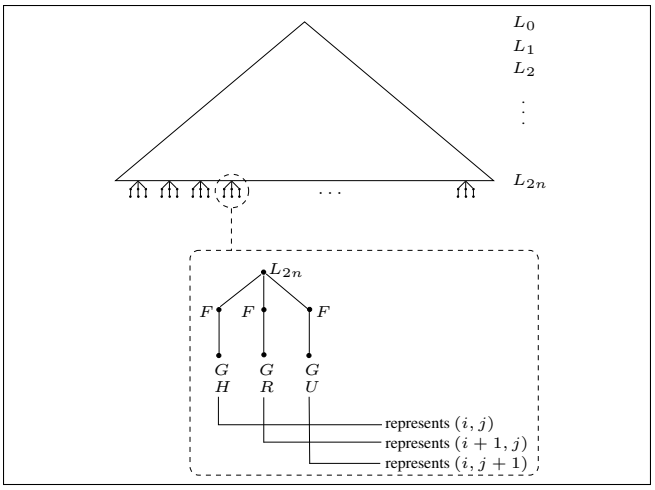

Figure 1: Structure of torus trees.

2. concept names $T_0, \ldots, T_k$ which represent tile types;
3. concept names H, R, U which stand for "here", "right", "up";
4. concept names $L_0, \ldots, L_{2n}$ to identify the levels of torus trees and concept names F and G to identify certain other nodes;
5. the role name r used in the composition S.

We refer to numbers between 0 and $2^{2n} - 1$ as a *grid position*: in its binary representation, bits 0 to $n-1$ represent the horizontal position in the grid and bits n to $2n-1$ the vertical position.

The next step is to define the TBox $\mathcal{T}_c$. We first give a few more details about torus trees, illustrated in Figure 1. There is binary branching on levels 0 to $2n-1$ and, intuitively, nodes on levels 0 to $2n$ form the torus tree proper while nodes on levels $2n+1$ and $2n+2$ form gadgets appended to the tree nodes on level $2n$. Such a gadget is highlighted in Figure 1. All nodes on level $2n+2$ are labeled with the concept name G, all nodes on level $2n+1$ with the concept name F, and all nodes on levels $i = 0..2n$ with the concept name L_i. Moreover, every L_{2n}-node is associated with a grid position via the concept names $A_i, \overline{A}_i$ (not shown in the figure). The G-node leaf below it that is labeled H is associated with the same position as its L_{2n}-node ancestor. In contrast, the R-leaf is associated with the neighboring position to the right and the U-leaf with the neighboring position to the top (all via $A_i, \overline{A}_i$). Every G-node is labeled with a tile T_i (not shown in the figure) such that the tiles of H- and R-nodes in the same gadget satisfy the horizontal matching condition, and likewise for the H- and U-node and the vertical matching condition. For technical reasons related to the query construction, the F-node is labeled complementarily regarding the concept names $A_i, \overline{A}_i$ compared to its G-node successor. Note that, so far, we have only required that the matching conditions are satisfied locally in each gadget. To ensure that a torus tree represents a solution, we will enforce later using the query q_c that whenever two G-nodes represent the same position, then they are labeled with the same tile.

We now construct the TBox $\mathcal{T}_c$. The last level of torus trees must be identified by the concept name G. Proper verification of that level is indicated by the concept name $G\mathsf{ok}$ (which is not in Σ_c):

$$A_i \sqsubseteq \mathsf{ok}_i \qquad \overline{A}_i \sqsubseteq \mathsf{ok}_i \qquad T_j \sqsubseteq T\mathsf{ok}$$
$$\mathsf{ok}_0 \sqcap \cdots \sqcap \mathsf{ok}_{2n-1} \sqcap T\mathsf{ok} \sqcap G \sqsubseteq G\mathsf{ok}$$

where i ranges over $0..2n-1$. Note that, to receive a $G\mathsf{ok}$ label, a G-node must be labeled with at least one of A_i and $\overline{A}_i$ for each i, and by at least one concept name of the form T_j. We next verify F-nodes:

$$A_i \sqcap \exists S.(G\mathsf{ok} \sqcap \overline{A}_i) \sqsubseteq \mathsf{ok}'_i$$
$$\overline{A}_i \sqcap \exists S.(G\mathsf{ok} \sqcap A_i) \sqsubseteq \mathsf{ok}'_i$$
$$\mathsf{ok}'_0 \sqcap \cdots \sqcap \mathsf{ok}'_{2n-1} \sqcap F \sqsubseteq F\mathsf{ok}$$

where i ranges over $0..2n-1$. Note that we have not yet guaranteed that G-nodes make true at most one of A_i and $\overline{A}_i$ for each i. Moreover, the first two lines may speak about different S-successors. It is thus not clear that they achieve the intended complementary labeling. We fix these problems by adding the following inclusion:

$$\exists S^{2n+1}.(\exists S.(G \sqcap A_i) \sqcap \exists S.(G \sqcap \overline{A}_i)) \sqsubseteq C_{q_c}$$

where i ranges over $0..2n-1$, $\exists S^{\ell}.C$ denotes ℓ-fold quantification $\exists S.\cdots\exists S.C$, and C_{q_c} is an $\mathcal{ELI}$-concept to be defined later that will satisfy Point 2 of Lemma 43, that is, make the query q_c true at the root of the torus tree. To understand this, assume for example that there is a G-node labeled with both A_0 and $\overline{A}_0$. Then C_{q_c} will be made true at the root of the torus tree and thus the ABox is ruled out as a witness in $(*)$ above.

We now verify the existence of level $2n$ of the tree, identified by the concept name L_{2n}. Each L_{2n}-node needs to have three S-successors, all of them F-nodes, labeled with H, R, U, respectively. Moreover, it must be labeled with $A_i, \overline{A}_i$ to represent the same grid position as the H-leaf below:

$$A_i \sqcap \exists S.(F\mathsf{ok} \sqcap \exists S.(G\mathsf{ok} \sqcap H \sqcap A_i)) \sqsubseteq \mathsf{ok}''_i$$
$$\overline{A}_i \sqcap \exists S.(F\mathsf{ok} \sqcap \exists S.(G\mathsf{ok} \sqcap H \sqcap \overline{A}_i)) \sqsubseteq \mathsf{ok}''_i$$
$$\exists S.(F\mathsf{ok} \sqcap \exists S.(G\mathsf{ok} \sqcap R)) \sqsubseteq R\mathsf{ok}$$
$$\exists S.(F\mathsf{ok} \sqcap \exists S.(G\mathsf{ok} \sqcap U)) \sqsubseteq U\mathsf{ok}$$
$$\exists S^{2n}.(\exists S^2.(G \sqcap X \sqcap A_i) \sqcap \exists S^2.(G \sqcap X \sqcap \overline{A}_i)) \sqsubseteq C_{q_c}$$

where i ranges over $0..2n-1$ and X ranges over H, R, U. The last inclusion makes such that all H-leaves below a L_{2n}-node have the same labeling regarding $A_i, \overline{A}_i$, and likewise for all R-leaves and all U-leaves. We next verify that the the grid positions of the H, R, U-nodes below a level $2n$-node relate in the intended way. We start with copying up the grid positions from the R-leaf and the U-leaf, for convenience:

$$L_{2n} \sqcap \exists S^2.(G \sqcap X \sqcap A_i) \sqsubseteq A_i^X$$
$$L_{2n} \sqcap \exists S^2.((G \sqcap X \sqcap \overline{A}_i)) \sqsubseteq \overline{A}_i^X$$

where i ranges over $0..2n-1$ and X ranges over R, U. The following inclusions are then used to verify that the horizontal component of the R-node is incremented compared to the H-node:

$$A_0 \sqcap \cdots \sqcap A_{i-1} \sqcap \overline{A}_i \sqcap A_i^R \sqsubseteq \mathsf{ok}_{HRi}$$
$$A_0 \sqcap \cdots \sqcap A_{i-1} \sqcap A_i \sqcap \overline{A}_i^R \sqsubseteq \mathsf{ok}_{HRi}$$
$$\overline{A}_j \sqcap \overline{A}_i \sqcap \overline{A}_i^R \sqsubseteq \mathsf{ok}_{HRi}$$
$$\overline{A}_j \sqcap A_i \sqcap A_i^R \sqsubseteq \mathsf{ok}_{HRi}$$

where i ranges over $0..n$ and j over $0..i$. We can use similar inclusions setting concept names $\mathsf{ok}_{HRn}, \ldots, \mathsf{ok}_{HR2n-1}$ when the vertical component of the R-node is identical to that of the H-node, concept names $\mathsf{ok}_{HU0}, \ldots, \mathsf{ok}_{HUn-1}$ when the horizontal component of the U-node is identical to that of the H-node, and concept names $\mathsf{ok}_{HUn}, \ldots, \mathsf{ok}_{HU2n-1}$ when the vertical component of the U-node is incremented compared to the H-node. To make L_{2n} true, which identifies level $2n$-nodes, we require that all checks succeeded:

$$\mathsf{ok}_{HR0} \sqcap \cdots \sqcap \mathsf{ok}_{HRm-1} \sqcap$$
$$\mathsf{ok}_{HU0} \sqcap \cdots \sqcap \mathsf{ok}_{HUm-1} \sqcap$$
$$\mathsf{ok}''_0 \sqcap \cdots \sqcap \mathsf{ok}''_{2n-1} \sqcap$$
$$R\mathsf{ok} \sqcap U\mathsf{ok} \sqcap L_{2n} \sqsubseteq L_{2n}\mathsf{ok}.$$

To locally ensure the tiling conditions at L_{2n}-nodes, we put for all $(i,j) \notin H$ and all $(i,\ell) \notin V$:

$$\exists S^{2n}.(\exists S^2.(H \sqcap T_i) \sqcap \exists S^2.(R \sqcap T_j)) \sqsubseteq C_{q_c}$$
$$\exists S^{2n}.(\exists S^2.(H \sqcap T_i) \sqcap \exists S^2.(U \sqcap T_\ell)) \sqsubseteq C_{q_c}.$$

We next verify the existence of levels $2n-1$ to 0 of the tree. To make sure that the required successors are present on all levels, we branch on the concept names A_i, $\overline{A}_i$ at level i and for all $j < i$, keep our choice of A_j, $\overline{A}_j$:

$$\exists S.(L_{i+1}\mathsf{ok} \sqcap A_i) \sqcap \exists S.(L_{i+1}\mathsf{ok} \sqcap \overline{A}_i) \sqsubseteq \mathsf{succ}_i$$
$$A_j \sqcap \exists S.(L_{i+1}\mathsf{ok} \sqcap A_j) \sqsubseteq \mathsf{ok}_{i,j}$$
$$\overline{A}_j \sqcap \exists S.(L_{i+1}\mathsf{ok} \sqcap \overline{A}_j) \sqsubseteq \mathsf{ok}_{i,j}$$
$$\mathsf{succ}_i \sqcap \mathsf{ok}_{i,0} \sqcap \cdots \sqcap \mathsf{ok}_{i,i-1} \sqcap L_i \sqsubseteq L_i\mathsf{ok}$$
$$\exists S^{i}.(\exists S.(L_{i+1} \sqcap A_j) \sqcap \exists S.(L_{i+1} \sqcap \overline{A}_j)) \sqsubseteq C_{q_c}$$

where i ranges over $0..2n-1$ and j over $0..i-1$. The initial condition is verified at the G-nodes. Put

$$\exists S^{2n+2}.(\overline{A}_0 \sqcap \cdots \sqcap \overline{A}_{2n-1} \sqcap T_i) \sqsubseteq C_{q_c}$$

for all $i \in \{0, \ldots, k\}$ with $i \neq c_0$, and similarly for the grid positions $(1,0), \ldots, (n-1,0)$.

We next define the query q_c to ensure that all G-nodes that are associated with the same grid position are labeled with the same tile type. A bit more verbosely, we have to guarantee that

$(*)$ if a and b are G-nodes labeled identically regarding the concept names $A_i, \overline{A}_i$, then there are no distinct tile types k, j such that a is labeled with T_k and b with T_j.

The UCQ $q_c^\vee$ contains one CQ for each choice of tile types k, j. Fix concrete such k, j. We construct the required CQ q from component queries $p_0, \ldots, p_{n-1}$, which all take the form of the query show on the left-hand side of Figure 2. Note that all

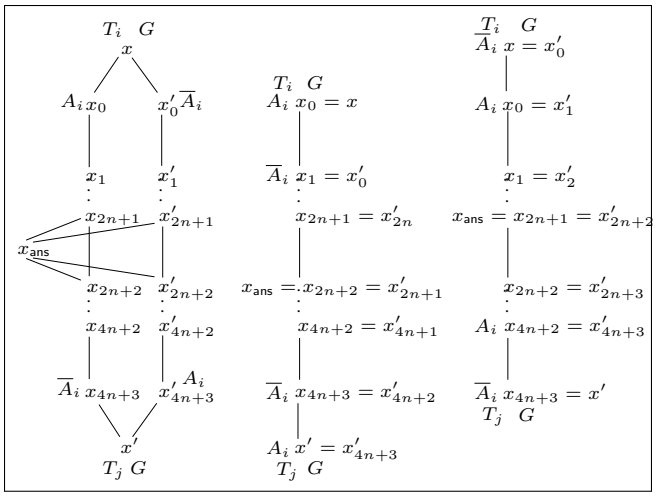

Figure 2: The query p_i (left) and two of its identifications (middle and right).

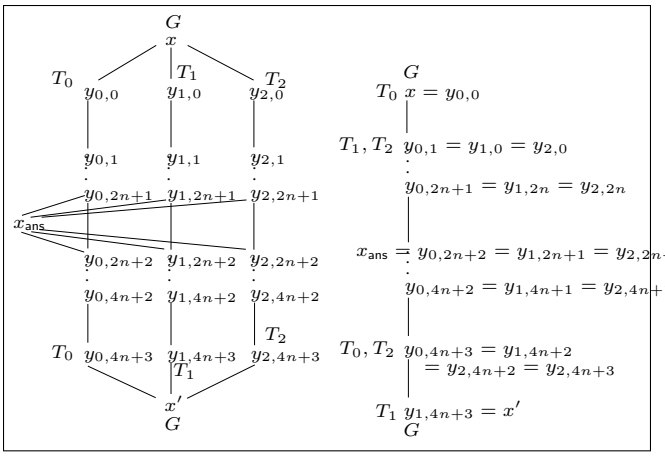

Figure 3: The query q_{tile} (left) and one of its identifications (right).

edges are S-edges, the only difference between the component queries is which concept names A_i and $\overline{A}_i$ are used, and x_{ans} is the only answer variable. We assemble $p_0, \dots, p_{n-1}$ into the desired query $q_c^\vee$ by taking variable disjoint copies of $p_0, \dots, p_{n-1}$ and then identifying (i) the x-variables of all components and (ii) the x'-variables of all components.

To see why $q_c^\vee$ achieves $(*)$, first note that the variables x and x' must be mapped to leaves of the torus tree because of their G-label. Call these leaves a and a'. Since x_0 and x_0' are connected to x in the query, both must then be mapped either to a or to its predecessor; likewise, x_{4n+3} and x'_{4n+3} must be mapped either to a' or to its predecessor. Because of the labeling of a and a' and the predecessors in the torus tree with A_i and $\overline{A}_i$, we are actually even more constrained: exactly one of x_0 and x_0' must be mapped to a, and exactly one of x_{4n+3} and x'_{4n+3} to a'. If x_0 is mapped to a, then x_{ans} must be identified with x_{2n+2} because as an answer variable it has to be mapped to the root of the tree and the only other option (identifying x_{ans} with x_{2n+1}) would thus require a path of length $2n+1$ between a and the root. Also for path length reasons, this means that x_{4n+3} must be mapped to the predecessor of a', thus x'_{4n+3} is mapped to a'. Analogously, we can show that mapping x_0' to a requires mapping x_{4n+3} to a'. These two options give rise to the two variable identifications in each query p_i shown in Figure 2. Note that the first case implies that a and a' are both labeled with A_i while they are both labeled with $\overline{A}_i$ in the second case. In summary, a and a' must thus agree on all concept names A_i, $\overline{A}_i$. Since a must satisfy T_i and a' must satisfy T_j due to the labeling of x and x', we have achieved $(*)$.

We now show how to replace the UCQ $q_c^\vee$ with a single CQ q_c. This requires the following changes:

1. the F-nodes in configuration trees receive additional labels: when a G-node is labeled with T_i, then its predecessor F-node is labeled with T_j for all $j \neq i$;
2. the query construction is modified.

Point 1 is important for the CQ to be constructed to work correctly and can be achieved in a straightforward way by modifying $\mathcal{T}_c$, details are omitted. We thus concentrate on Point 2. The desired CQ q_c is again constructed from component queries. We use n components as shown in Figure 2, except that the T_i- and T_j-labels are dropped. We further add the component shown in Figure 3 where again x and x' are the variables shared with the other components, and where we assume for simplicity that $T = \{0, 1, 2\}$; the generalization to an unrestricted number of tile types is straightforward, see [Lutz, 2007]. The additional component can be understood essentially in the same way as the previous query components.

Lemma 46. *$\mathcal{T}_c$, q_c, and Σ_c satisfy Points 1 and 2 from Lemma 43 when choosing $A^* = L_0\mathsf{ok}$.*

Proof. (sketch) We show the following:

1. If P has a solution given c, then there is a Σ_c-ABox $\mathcal{A}$ and an $a \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{T}_c \models A^*(a)$ and $\mathcal{A}, \mathcal{T}_c \not\models q_c(a)$.
2. If P has no solution given c, then for any Σ_c-ABox $\mathcal{A}$ and $a \in \mathsf{Ind}(\mathcal{A})$, $\mathcal{A}, \mathcal{T}_c \models A^*(a)$ implies $\mathcal{A}, \mathcal{T}_c \models q_c(a)$.
3. There is an $\mathcal{ELI}$-concept C_{q_c} such that $d \in C^{\mathcal{I}}$ implies $\mathcal{I} \models q_c(d)$.
4. q_c is FO-rewritable relative to $\mathcal{T}_c$ and Σ_c.

(1) Take as $\mathcal{A}$ a torus tree that encodes a solution for P given c (viewed as an ABox) and let a be the root of the tree. The verification of torus trees by $\mathcal{T}_c$ yields $\mathcal{A}, \mathcal{T}_c \models L_0\mathsf{ok}(a)$. Since the torus tree is not defective, we have $\mathcal{A}, \mathcal{T}_c \not\models q_c(a)$.

(2) Since the verification of (homomorphic images of) torus trees by $\mathcal{T}_c$ is sound, $\mathcal{A}, \mathcal{T}_c \models L_0\mathsf{ok}(a)$ implies that $\mathcal{A}$ contains a homomorphic image of a torus tree whose root is identified by a. Since there is no solution for P given c, that tree must be defective. Consequently, $\mathcal{A}, \mathcal{T}_c \models q_c(a)$.

(3) Set

$$G_1 = G \sqcap \overline{A}_0 \sqcap \cdots \sqcap \overline{A}_k \sqcap T_0$$
$$G_2 = G \sqcap \overline{A}_0 \sqcap \cdots \sqcap \overline{A}_k \sqcap T_1$$
$$F_1 = A_0 \sqcap \cdots \sqcap A_n \sqcap T_1 \sqcap \cdots \sqcap T_k$$
$$F_2 = A_0 \sqcap \cdots \sqcap A_n \sqcap T_0 \sqcap T_2 \sqcap \cdots \sqcap T_k$$
$$C_{q_c} = \exists S^{2n+1}.(F_1 \sqcap \exists S.G_1) \sqcap \exists S^{2n+1}.(F_2 \sqcap \exists S.G_2)$$

It can be verified that C_{q_c} is as required.

(4) Note that whenever $D \sqsubseteq C_{q_c}$ is in $\mathcal{T}_c$, then D uses symbols from Σ_c, only. One can construct an FO-rewriting of q_c relative to $\mathcal{T}_c$ and Σ_c that has the form

$$q_0(x) \vee \bigvee_{D \sqsubseteq C_{q_c} \in \mathcal{T}_c} q_D(x)$$

where q_D is D viewed as a CQ. To define q_0, let $\mathcal{T}_c^0$ be the result of removing from $\mathcal{T}_c$ all CIs of the form $D \sqsubseteq C_{q_c}$. Note that the recursion depth of $\mathcal{T}_c^0$ is bounded by $2n+1$. We can thus choose

$$q_0(x) = \bigvee_{\mathcal{A} \in \mathfrak{A}} q_{\mathcal{A}}(x)$$

where $\mathfrak{A}$ is the set of all pseudo tree Σ_c-ABoxes $\mathcal{A}$ of depth at most $2n+1$, width at most $|q_c|$, outdegree at most $|\mathcal{T}_c|$, and with root a_0 such that $\mathcal{A}, \mathcal{T}_c \models q_c[a_0]$ and where $q_{\mathcal{A}}$ is $\mathcal{A}$ viewed as a CQ. □

E.3 2ExpTime lower bounds

We consider Boolean (connected) CQs. We reduce the word problem of exponentially space bounded alternating Turing machines (ATMs), see [Chandra *et al.*, 1981]. An *Alternating Turing Machine (ATM)* is of the form $M = (Q, \Sigma, \Gamma, q_0, \Delta)$. The set of *states* $Q = Q_\exists \uplus Q_\forall \uplus \{q_a\} \uplus \{q_r\}$ consists of *existential states* in $Q_\exists$, *universal states* in $Q_\forall$, an *accepting state* q_a, and a *rejecting state* q_r; Σ is the *input alphabet* and Γ the *work alphabet* containing a *blank symbol* $\Box$ and satisfying $\Sigma \subseteq \Gamma$; $q_0 \in Q_\exists \cup Q_\forall$ is the *starting* state; and the *transition relation* Δ is of the form

$$\Delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}.$$

We write $\Delta(q, \sigma)$ to denote $\{(q', \sigma', M) \mid (q, \sigma, q', \sigma', M) \in \Delta\}$ and assume w.l.o.g. that the state q_0 cannot be reached by any transition.

A *configuration* of an ATM is a word wqw' with $w, w' \in \Gamma^*$ and $q \in Q$. The intended meaning is that the one-side infinite tape contains the word ww' with only blanks behind it, the machine is in state q, and the head is on the symbol just after w. The *successor configurations* of a configuration wqw' are defined in the usual way in terms of the transition relation Δ. A *halting configuration* is of the form wqw' with $q \in \{q_a, q_r\}$.

A *computation* of an ATM M on a word w is a (finite or infinite) sequence of configurations $K_0, K_1, \ldots$ such that $K_0 = q_0 w$ and K_{i+1} is a successor configuration of K_i for all $i \geq 0$. The ATMs considered in the following have only *finite* computations on any input. Since this case is simpler than the general one, we define acceptance for ATMs with finite computations, only. Let M be such an ATM. A halting configuration is *accepting* iff it is of the form $wq_a w'$. For other configurations $K = wqw'$, acceptance depends on q: if $q \in Q_\exists$, then K is accepting iff at least one successor configuration is accepting; if $q \in Q_\forall$, then K is accepting iff all successor configurations are accepting. Finally, the ATM M with starting state q_0 *accepts* the input w iff the *initial configuration* $q_0 w$ is accepting. We use $L(M)$ to denote the language accepted by M.

There is an exponentially space bounded ATM M whose word problem is 2ExpTime-hard and we may assume that the length of every computation path of M on $w \in \Sigma^n$ is bounded by 2^{2^n}, and all the configurations wqw' in such computation paths satisfy $|ww'| \leq 2^n$, see [Chandra *et al.*, 1981].

Lemma 47. *Given an input w to M, one can construct in polynomial time an $\mathcal{ELI}$ TBox $\mathcal{T}_w$, a Boolean connected CQ q_w, and an ABox signature Σ_w such that, for a selected concept name $A^* \notin \Sigma_w$,*

1. *M accepts w iff there is a Σ_w-ABox $\mathcal{A}$ such that $\mathcal{A}, \mathcal{T}_w \models \exists x\, A^*(x)$ and $\mathcal{A}, \mathcal{T}_w \not\models q_w$;*
2. *M accepts w iff there is a Σ_w-ABox $\mathcal{A}$ and an $a \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{T}_w \models A^*(a)$ and $\mathcal{A}, \mathcal{T}_w \not\models q_w$;*
3. *q_w is FO-rewritable relative to $\mathcal{T}_w$ and Σ_w;*
4. *there is an $\mathcal{ELI}$-concept C_{q_w} such that $C_{q_w}^{\mathcal{I}} \neq \emptyset$ implies $\mathcal{I} \models q_w$.*

Theorem 48. *Containment in $(\mathcal{ELI}, \text{CQ})$ is 2ExpTime-hard.*

Proof. Let w be an input to M, $\mathcal{T}_w$, q_w, and Σ_w as in Lemma 47. By Point 1 of Lemma 47, $(\mathcal{T}_w, \Sigma_w, \exists x\, A^*(x)) \not\subseteq (\mathcal{T}_w, \Sigma_w, q_w)$ over Σ_w-ABoxes iff M accepts w. □

Theorem 49. *FO-rewritability in $(\mathcal{ELI}, \text{CQ})$ is 2ExpTime-hard.*

Proof. Let w be an input to M and $\mathcal{T}_w$, q_w, Σ_w as in Lemma 47. We obtain a TBox $\mathcal{T}$ by extending $\mathcal{T}_w$ with the following:

$$\begin{aligned} \exists r.A &\sqsubseteq A \\ A \sqcap B \sqcap A^* &\sqsubseteq C_{q_w} \end{aligned}$$

where A, B, and r do not occur in $\mathcal{T}_w$ and q_w, $A^* \notin \Sigma_w$ is the concept name from Lemma 47 and C_{q_w} the concept from Point 4 of that lemma. Set $\Sigma = \Sigma_w \cup \{A, B, r\}$. It remains to prove the following.

Claim. M accepts w iff q_w is not FO-rewritable relative to $\mathcal{T}$ and Σ.

First assume that M accepts w. By Point 2 of Lemma 47, there is a Σ_w-ABox $\mathcal{A}$ and $a_0 \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{T}_w \models A^*(a_0)$ and $\mathcal{A}, \mathcal{T}_w \not\models q_w$. Since every $\mathcal{ELI}$ TBox is unraveling tolerant [Lutz and Wolter, 2012] and by compactness, we can assume w.l.o.g. that $\mathcal{A}$ is tree-shaped with root a_0. Let ℓ be the depth of $\mathcal{A}$. For each $k > \ell$, let $\mathcal{A}_k$ be the ABox obtained by extending $\mathcal{A}$ with

$$B(a_0), r(a_0, a_1), \ldots, r(a_{k-1}, a_k), A(a_k)$$

where $a_k, \ldots, a_1$ do not occur in $\mathcal{A}$. Note that $\mathcal{A}_k$ is tree-shaped and of depth at least k. Since $\mathcal{A}, \mathcal{T}_w \models A^*(a_0)$, we have $\mathcal{A}_k, \mathcal{T} \models C_{q_w}(a_0)$. Applying Point 4 of Lemma 47, we

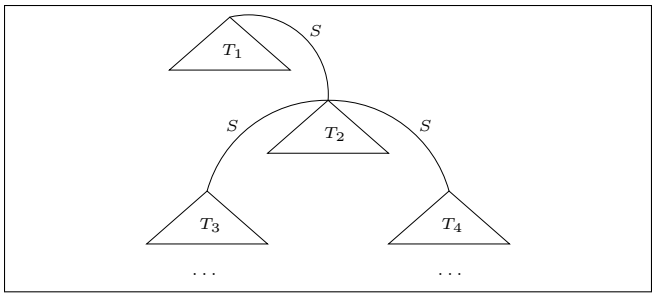

Figure 4: Representing ATM computations.

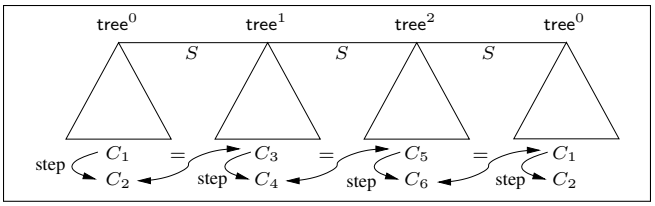

Figure 5: Representing ATM computations.

obtain $\mathcal{A}_k, \mathcal{T} \models q_w$. To prove that q_w is not FO-rewritable relative to $\mathcal{T}$ and Σ, by Theorem 9 it suffices to show that $\mathcal{A}|_{>0}, \mathcal{T} \not\models q_w$ and $\mathcal{A}_k|_{\leq k-1}, \mathcal{T} \not\models q_w$. Note that neither $\mathcal{A}_k|_{>0}$ nor $\mathcal{A}_k|_{\leq k-1}$ contains an r-path from an individual satisfying B to an individual satisfying A. On such ABoxes, $\mathcal{T}$ can be replaced with $\mathcal{T}_w$ since the left-hand sides of the second additional concept inclusions in $\mathcal{T}$ will never apply, and that concept inclusion is the only (additional) one whose right-hand side contains symbols from $\mathcal{T}$ and q_w. It thus suffices to show that $\mathcal{A}_k|_{>0}, \mathcal{T}_w \not\models q_w$ and $\mathcal{A}_k|_{k-1}, \mathcal{T}_w \not\models q_w$. This follows from $\mathcal{A}, \mathcal{T}_w \not\models q_w$ and the fact that A, B, and r (the only symbols in assertions from $(\mathcal{A}_k|_{>0}) \setminus \mathcal{A}$ and $(\mathcal{A}_k|_{k-1}) \setminus \mathcal{A}$) occur neither in $\mathcal{T}_w$ nor in q_w.

Now assume that M does not accept w. By Point 3 of Lemma 47, there is an FO-rewriting $\widehat{q}_w$ of q_w relative to $\mathcal{T}_w$ and Σ_w. We argue that $\widehat{q}_w$ is also an FO-rewriting of q_w relative to $\mathcal{T}$ and Σ.

First assume $\mathcal{A} \models \widehat{q}_w$ for some Σ-ABox $\mathcal{A}$. Since $\widehat{q}_w$ uses only symbols from Σ_w, this means that $\mathcal{A}' \models \widehat{q}_w$ where $\mathcal{A}'$ is the reduct of $\mathcal{A}$ to symbols in Σ_w. Thus $\mathcal{A}', \mathcal{T}_w \models q_w$, implying $\mathcal{A}, \mathcal{T} \models q_w$.

Conversely, assume that $\mathcal{A}, \mathcal{T} \models q_w$ for some Σ-ABox $\mathcal{A}$. Using canonical models and the construction of $\mathcal{T}$, one can show that this implies (i) $\mathcal{A}, \mathcal{T}_w \models q_w$ or (ii) $\mathcal{A}, \mathcal{T}_w \models \exists x\, A^*(x)$. In Case (i), we get $\mathcal{A}', \mathcal{T}_w \models q_w$, where $\mathcal{A}'$ is the Σ_w-reduct of $\mathcal{A}$. Thus $\mathcal{A}' \models \widehat{q}_w$, which implies $\mathcal{A} \models \widehat{q}_w$. In Case (ii), Point 1 of Lemma 47 yields $\mathcal{A}, \mathcal{T}_w \models q_w$ and thus we can proceed as in Case (i). □

E.4 Proof of Lemma 47

Let $w = \sigma_0 \cdots \sigma_{m-1} \in \Sigma^*$ be an input to M. We show how to construct a TBox $\mathcal{T}_w$, query q_w, and ABox signature Σ_w that satisfy Points 1 to 4 of Lemma 47. We first use a UCQ for q_w, which results in a simpler reduction, and in a second step show how to replace the UCQ with a CQ.

In the reduction, we represent each configuration of a computation of M by the leaves of a *configuration tree* that has depth $n + 2$ and whose edges are represented by the role composition $S = r^-; r$, similarly to the representation of the $2^n \times 2^n$-torus in the previous reduction. The trees representing configurations are then interconnected to a *computation tree* which represents the computation of M on w. This is illustrated in Figure 4, where the tree T_1 represents an existential configuration and thus has only one successor tree T_2, connected via the same role composition S that is also used inside configuration trees. In contrast, T_2 represents a universal configuration with two successor configurations T_3 and T_4.

The above description is actually an oversimplification. In fact, every configuration tree stores two configurations instead of only one: the current configuration and the previous configuration in the computation. The query q_w to be defined later on makes sure that the previous configuration stored in a configuration tree is identical to the current configuration stored in its predecessor configuration tree. The actual transitions of M are then represented locally inside configuration trees. This is illustrated by a sequence of existential configurations in Figure 5 where each C_i represents a stored configuration, "step" denotes a transition of M, and "=" denotes identity of stored configurations.

Since the role composition S used to connect configuration trees is symmetric, it is difficult to distinguish predecessor configuration trees from successor configuration trees. To break this symmetry, we represent the current and next configuration stored in configuration trees using six different sets of concept names. This is also indicated in Figure 5 where C_i means that we use the i-th set of concept names for representing the stored configuration.

We next construct the TBox $\mathcal{T}_w$, which is used to verify the existence of an accepting computation tree of M on input w in the ABox, apart from the described copying of stored configurations which will be achieved by the query q_w later on. The ABox signature Σ_w consists of the following symbols:

1. concept names $A_0, \ldots, A_{n-1}$ and $\overline{A}_0, \ldots, \overline{A}_{n-1}$ that serve as bits in the binary representation of a number between 0 and $2^n - 1$, identifying the position of tape cells (that is, leaves in configuration trees);
2. for each $\sigma \in \Gamma$, the concept names A^i_σ, $1 \leq i \leq 6$;
3. for each $\sigma \in \Gamma$ and $q \in Q$, the concept names $A^i_{q,\sigma}$, $1 \leq i \leq 6$;
4. the concept names $\overline{H}$, W, and $\overline{W}$ which stand for "cell without head", "cell being written to reach current configuration", and "cell not being written to reach current configuration";
5. a concept name $A_{q,\sigma,M}$ for each $q \in Q$, $\sigma \in \Gamma$, and $M \in \{L, R\}$ to describe transitions of M;
6. a concept name I that marks the initial configuration;
7. concept names $L_0, \ldots, L_n$ to identify the levels of configuration trees and concept names F_1, F_2, G_1, G_2 to identify certain other nodes;
8. the role name r used in the composition S.

The concept names A^i_σ are used to represent the symbols on the tape that are currently not under the head and $A^i_{q,\sigma}$ to mark tape cells under the head, indicating the head position, the current state, and the symbol under the head.

We start with verifying single configuration trees. Such trees come in three different types, depending on the set of concept names that we use to represent the current and previous configuration stored. This is shown in Figure 5. Type 0 means that the previous configuration is represented by concept names of the form A^1_σ and $A^1_{q,\sigma}$ and the current configuration by concept names A^2_σ and $A^2_{q,\sigma}$, type 1 uses A^3_σ and $A^3_{q,\sigma}$ for the previous configuration, and so on. We start with verifying configuration trees of type 0. Intuitively, nodes on levels 0 to n form the configuration tree proper while nodes on levels $n+1$ and $n+2$ form gadgets appended to the tree nodes on level n, similarly to what is shown in Figure 1. We identify each node on level $n+1$ with one of the concept names F_1, F_2 and each node on level $n+2$ with one of the concept names G_1, G_2. In contrast to Figure 1, there are only two nodes below each level n node, one labeled F_1 and one labeled F_2. Moreover, every F_ℓ node must have a G_ℓ-node successor. Each G_ℓ-node represents a tape cell, and the position of that cell is encoded in binary by the concept names $A_i, \overline{A}_i$. The G_1- and G_2-node below the same level n node must both have the same position and, for similar reasons as in the previous reduction, the F_ℓ nodes in between receive a complementary labeling regarding these concept names and also regarding the concept names $A^\ell_\sigma, A^\ell_{q,\sigma}$. At G_1-nodes, the concept names A^1_σ and $A^1_{q,\sigma}$ are used to store information and at G_2-nodes, we use the concept names A^2_σ and $A^2_{q,\sigma}$. Thus, G_1-nodes represent the previous configuration while G_2-nodes representing the current configutation. The concept names $\overline{H}, W, \overline{W}$ are used for the current configuration, only.

The verification of configuration trees is again bottom-up, starting at level $n+2$ nodes:

$$A_i \sqsubseteq \mathsf{ok}_i \qquad \overline{A}_i \sqsubseteq \mathsf{ok}_i \qquad A^1_\sigma \sqsubseteq \Gamma\mathsf{ok}_1 \quad A^1_{q,\sigma} \sqsubseteq \Gamma\mathsf{ok}_1$$
$$\mathsf{ok}_0 \sqcap \cdots \sqcap \mathsf{ok}_{n-1} \sqcap \Gamma\mathsf{ok}_1 \sqcap G_1 \sqsubseteq G_1\mathsf{ok}$$
$$A^2_\sigma \sqcap \overline{H} \sqcap \overline{W} \sqsubseteq \Gamma\mathsf{ok}_2 \quad A^2_\sigma \sqcap \overline{H} \sqcap W \sqsubseteq \Gamma\mathsf{ok}_2$$
$$A^2_{q,\sigma} \sqcap \overline{W} \sqsubseteq \Gamma\mathsf{ok}_2$$
$$\mathsf{ok}_0 \sqcap \cdots \sqcap \mathsf{ok}_{n-1} \sqcap \Gamma\mathsf{ok}_2 \sqcap G_2 \sqsubseteq G_2\mathsf{ok}$$

where i ranges over $0..n-1$, q over the elements of Q and σ over the elements of Γ. Note that every G_ℓ-node must be labeled with at least one of A_i and $\overline{A}_i$ for each i and by at least one concept name of the form A^ℓ_σ or $A^\ell_{q,\sigma}$. If $\ell = 2$, then an A^ℓ_σ-label (as opposed to an $A^\ell_{q,\sigma}$-label) is acceptable only if there is also an $\overline{H}$-label. For $\ell = 2$, there must also be a W- or $\overline{W}$-label, the former only being acceptable if the head is not on the current cell. We now verify F_ℓ-nodes:

$$A_i \sqcap \exists S.(G_\ell\mathsf{ok} \sqcap \overline{A}_i) \sqsubseteq \mathsf{ok}_{\ell,i}$$
$$\overline{A}_i \sqcap \exists S.(G_\ell\mathsf{ok} \sqcap A_i) \sqsubseteq \mathsf{ok}_{\ell,i}$$
$$\bigsqcap_{\beta \in (\Gamma \cup (Q\times\Gamma))\setminus\{\alpha\}} A^\ell_\beta \sqcap \exists S.(G_\ell\mathsf{ok} \sqcap A^\ell_\alpha) \sqsubseteq \Gamma\mathsf{ok}'_\ell$$
$$\mathsf{ok}_{\ell,0} \sqcap \cdots \sqcap \mathsf{ok}_{\ell,n-1} \sqcap \Gamma\mathsf{ok}'_\ell \sqcap F_\ell \sqsubseteq F_\ell\mathsf{ok}$$

where ℓ ranges over 1,2, i over $0..n-1$, and α over $\Gamma \cup (Q\times\Gamma)$. We have not yet guaranteed that G_ℓ-nodes make true at most one of A_i and $\overline{A}_i$ for each i, at most one concept name of the form A^ℓ_α, and not simultaneously W and $\overline{W}$, or $\overline{H}$ and a concept name of the form $A^2_{q,\sigma}$, or W and a concept name $A^2_{q,\sigma}$. Moreover, the first three lines may speak about different S-successors. It is thus not clear that they achieve the intended complementary labeling. We fix these problems by adding the following inclusions:

$$\exists S.(G_\ell \sqcap A_i) \sqcap \exists S.(G_\ell \sqcap \overline{A_i}) \sqsubseteq C_{q_w}$$
$$\exists S.(G_\ell \sqcap A^\ell_\alpha) \sqcap \exists S.(G_\ell \sqcap A^\ell_\beta) \sqsubseteq C_{q_w}$$
$$\exists S.(G_\ell \sqcap W) \sqcap \exists S.(G_\ell \sqcap \overline{W}) \sqsubseteq C_{q_w}$$
$$\exists S.(G_\ell \sqcap A^2_{q,\sigma}) \sqcap \exists S.(G_\ell \sqcap \overline{H}) \sqsubseteq C_{q_w}$$
$$\exists S.(G_\ell \sqcap A^2_{q,\sigma}) \sqcap \exists S.(G_\ell \sqcap W) \sqsubseteq C_{q_w}$$

where ℓ ranges over $1, 2$, i over $0..n-1$, α, β take distinct values from $\Gamma \cup (Q \times \Gamma)$, q ranges over Q, and σ over Γ. Moreover, C_{q_w} is an $\mathcal{ELI}$-concept to be defined later that will satisfy Point 4 of Lemma 47, that is, make the query q_w true.

We now verify the existence of level n of the tree, identified by the concept name L_n. Nodes here need to have S-successors in F_1 and F_2 that are again labeled complementarily regarding the concept names $A_i, \overline{A}_i$ (in other words, the L_n node agrees with the labeling of the G_1- and G_2-node leaves below it):

$$A_i \sqcap \exists S.(F_\ell\mathsf{ok} \sqcap \overline{A}_i) \sqsubseteq \mathsf{ok}'_{\ell,i}$$
$$\overline{A}_i \sqcap \exists S.(F_\ell\mathsf{ok} \sqcap A_i) \sqsubseteq \mathsf{ok}'_{\ell,i}$$
$$\mathsf{ok}'_{1,0} \sqcap \cdots \sqcap \mathsf{ok}'_{1,n-1} \sqcap \mathsf{ok}'_{2,0} \sqcap \cdots \sqcap \mathsf{ok}'_{2,n-1} \sqcap L_n \sqsubseteq L_n\mathsf{ok}$$
$$\exists S.(F_\ell \sqcap A_i) \sqcap \exists S.(F_\ell \sqcap \overline{A}_i) \sqsubseteq C_{q_w}$$
$$\exists S.(F_\ell \sqcap A^\ell_\alpha) \sqcap \exists S.(F_\ell \sqcap A^\ell_\beta) \sqsubseteq C_{q_w}$$

where ℓ ranges over $1..2$, i over $0..n-1$, and α, β take distinct values from $\Gamma \cup (Q \times \Gamma)$.

We next verify the existence of levels $n-1$ to 0 of the configuration tree. We exploit that we have already stored the position of the leaves in the concept names $A_i, \overline{A}_i$ at L_n-nodes. Each node on level i branches on the concept names $A_i, \overline{A}_i$ and keeps the choice of $A_j, \overline{A}_j$ for all $j < i$:

$$\exists S.(L_{i+1}\mathsf{ok} \sqcap A_i) \sqcap \exists S.(L_{i+1}\mathsf{ok} \sqcap \overline{A}_i) \sqsubseteq \mathsf{succ}_i$$
$$A_j \sqcap \exists S.(L_{i+1}\mathsf{ok} \sqcap A_j) \sqsubseteq \mathsf{ok}''_{i,j}$$
$$\overline{A}_j \sqcap \exists S.(L_{i+1}\mathsf{ok} \sqcap \overline{A}_j) \sqsubseteq \mathsf{ok}''_{i,j}$$
$$\mathsf{succ}_i \sqcap \mathsf{ok}''_{i,0} \sqcap \cdots \sqcap \mathsf{ok}''_{i,i-1} \sqcap L_i \sqsubseteq L_i\mathsf{ok}$$
$$\exists S.(L_{i+1} \sqcap A_j) \sqcap \exists S.(L_{i+1} \sqcap \overline{A}_j) \sqsubseteq C_{q_w}$$

where i ranges over $0..n-1$ and j over $0..i-1$. We also want that configuration trees have exactly one leaf labeled with a concept name of the form $A^2_{q,\sigma}$ and exactly one leaf labeled with W. We start with enforcing the "at most one" part of

"exactly one":

$$F_2 \sqcap \exists S.(G_2 \sqcap A^2_{q,\sigma}) \sqsubseteq H$$
$$L_n \sqcap \exists S.(F_2 \sqcap H) \sqsubseteq H$$
$$L_i \sqcap \exists S.(L_{i+1} \sqcap H) \sqsubseteq H$$
$$L_i \sqcap \exists S.(L_{i+1} \sqcap A_i \sqcap H) \sqcap \exists S.(L_{i+1} \sqcap \overline{A}_i \sqcap H) \sqsubseteq C_{q_w}$$
$$F_2 \sqcap \exists S.(G_2 \sqcap W) \sqsubseteq W'$$
$$L_n \sqcap \exists S.(F_2 \sqcap W') \sqsubseteq W'$$
$$L_i \sqcap \exists S.(L_{i+1} \sqcap W') \sqsubseteq W'$$
$$L_i \sqcap \exists S.(L_{i+1} \sqcap A_i \sqcap W') \sqcap \exists S.(L_{i+1} \sqcap \overline{A}_i \sqcap W') \sqsubseteq C_{q_w}$$

where i ranges over $0..n-1$ and q, σ over $Q \times \Gamma$. Note that we use A_i and $\overline{A}_i$ to distinguish left successors and right successors in the tree: when we see a label $A^2_{q,\sigma}$ at a G_2-leaf, we propagate the marker H up the tree and additionally make sure that, at no node of the tree, we have an H-marker coming both from the left successor and from the right successor. We deal with W in a similar way, propagating the marker W'.

The "at least one" part of "exactly one" requires some changes to the concept inclusions already given, which we only sketch. We have not included theses changes in the original version of the inclusions above to avoid cluttering the presentation. Essentially, we have to keep track of where we have already seen a concept name of the form $A^2_{q,\sigma}$ in a G_2-leaf and where we have already seen a G_2-leaf labeled W. For simplicity, let us concentrate on the latter. We replace the concept inclusion

$$\mathsf{ok}_{2,0} \sqcap \cdots \sqcap \mathsf{ok}_{2,n-1} \sqcap \Gamma\mathsf{ok}'_2 \sqcap F_2 \sqsubseteq F_2\mathsf{ok}$$

above with

$$\mathsf{ok}_{2,0} \sqcap \cdots \sqcap \mathsf{ok}_{2,n-1} \sqcap \Gamma\mathsf{ok}'_2 \sqcap F_2 \sqcap \exists S.(G_2 \sqcap W) \sqsubseteq F_2^W\mathsf{ok}$$
$$\mathsf{ok}_{2,0} \sqcap \cdots \sqcap \mathsf{ok}_{2,n-1} \sqcap \Gamma\mathsf{ok}'_2 \sqcap F_2 \sqcap \exists S.(G_2 \sqcap \overline{W}) \sqsubseteq F_2^{\overline{W}}\mathsf{ok}$$

Note that we have replaced $F_2\mathsf{ok}$ in the conclusion with $F_2^W\mathsf{ok}$ and $F_2^{\overline{W}}\mathsf{ok}$, recording whether or not there is a G_2-node satisfying W below. The information that we have seen W is then propagated propagated up the tree, which requires replacing each of $L_n\mathsf{ok}, \ldots, L_1\mathsf{ok}$ with two versions, $L_i^W\mathsf{ok}$ and $L_i^{\overline{W}}\mathsf{ok}$. On each level, we set $L_i^{\overline{W}}\mathsf{ok}$ if both successors are labeled with $L_{i+1}^{\overline{W}}\mathsf{ok}$ and $L_i^W\mathsf{ok}$ if one successor is labeled with $L_{i+1}^W\mathsf{ok}$, but not both. In fact, if both successors are labeled $L_{i+1}^W\mathsf{ok}$, then neither $L_i^W\mathsf{ok}$ nor $L_i^{\overline{W}}\mathsf{ok}$ will be set and this is exactly how we can ensure that there is at most one G_2-leaf labeled W. It can be enforced in an analogous way that there is a G_2 labeled with a concept name of the form $A^2_{q,\sigma}$. In fact, we have to deal with both W and these concept names simultaneously, using concept names such as $L_i^{W,H}\mathsf{ok}$ indicating that we are at a tree node on level i below which there is a G_2-leaf satisfying W and a G_2-leaf satisfying a concept name $A^2_{q,\sigma}$. Details are omitted.

At this point, we have essentially finished the verification of configuration trees of type 0 (we will comment on the other types below) and move on to verify computation trees, also in a bottom-up fashion. To be a proper part of a computation tree, a configuration must describe an accepting halting configuration or have successor configuration trees as required by the transition relation. For type 0 configuration trees, the former case is covered by

$$L_0\mathsf{ok} \sqcap \exists S^{n+2}.(G_2 \sqcap A^2_{q_a,\sigma}) \sqsubseteq \mathsf{tree}_0$$
$$L_0 \sqcap \exists S^{n+2}.(G_2 \sqcap A^2_\alpha) \sqcap \exists S^{n+2}.(G_2 \sqcap A^2_\beta) \sqsubseteq C_{q_w}$$

where q_a is the accepting state, σ ranges over all elements of Γ, and α and β are distinct elements of $Q \times \Gamma$. For the latter case and existential states, we add

$$L_0\mathsf{ok} \sqcap \exists S^{n+2}.(G_2 \sqcap A^2_{q_\exists,\sigma_0}) \sqcap \exists S.(\mathsf{tree}_1 \sqcap A_{q_1,\sigma_1,M_1}) \sqsubseteq \mathsf{tree}_0$$

for all $q_\exists \in Q_\exists$, $\sigma_0 \in \Gamma$, and $(q_1, \sigma_1, M_1) \in \Delta(q_\exists, \sigma_0)$; for universal states, we add

$$L_0\mathsf{ok} \sqcap \exists S^{n+2}.(G_2 \sqcap A^2_{q_\forall,\sigma_0}) \sqcap \exists S.(\mathsf{tree}_1 \sqcap A_{q_1,\sigma_1,M_1}) \sqcap \cdots \sqcap \exists S.(\mathsf{tree}_1 \sqcap A_{q_k,\sigma_k,M_k}) \sqsubseteq \mathsf{tree}_0$$

for all $q_\forall \in Q_\forall$ and $\sigma_0 \in \Gamma$ when $\Delta(q_\forall, \sigma_0) = \{(q_1, \sigma_1, M_1), \ldots, (q_k, \sigma_k, M_k)\}$. Note that we have used the concept names $A_{q,\sigma,M}$ as markers here. We still need to enforce that they really represent the transition in the configuration tree at whose root they are located. We do this as follows. Each marker state is the actual state in the current configuration:

$$A_{q_1,\sigma_1,M} \sqcap \exists S^{n+2}.(G_2 \sqcap A^2_{q_2,\sigma_2}) \sqsubseteq C_{q_w}$$

for all distinct $q_1, q_2 \in Q$, all $\sigma_1, \sigma_2 \in \Gamma$, and all $M \in \{L, R\}$. Each marker symbol is the actual symbol written in the current configuration:

$$A_{q_1,\sigma_1,M} \sqcap \exists S^{n+2}.(G_2 \sqcap W \sqcap A^2_{\sigma_2}) \sqsubseteq C_{q_w}$$

for all distinct $\sigma_1, \sigma_2 \in \Gamma$, all $q_1 \in Q$ and all $M \in \{L, R\}$. Each marker movement is the actual movement in the current configuration. To achieve this, we first say that the W-marker is exactly where the head was before:

$$L_n \sqcap \exists S^2.(G_1 \sqcap A^1_{q,\sigma}) \sqcap \exists S^2.(G_2 \sqcap \overline{W}) \sqsubseteq C_{q_w}$$

for all $q \in Q$ and $\sigma \in \Sigma$. Now, right moves are ensured in the following way:

$$A_{q,\sigma,R} \sqcap \exists S^i.[L_i \sqcap \exists S.(L_{i+1} \sqcap \overline{A}_i \sqcap \exists S.(L_{i+2} \sqcap A_i \sqcap \exists S.\cdots \sqcap \exists S.(L_n \sqcap A_n \sqcap \exists S^2.(G_2 \sqcap \overline{W})\cdots) \sqcap \exists S.(L_{i+1} \sqcap A_i \sqcap \exists S.(L_{i+2} \sqcap \overline{A}_i \sqcap \exists S.\cdots \sqcap \exists S.(L_n \sqcap \overline{A}_n \sqcap \exists S^2.(G_2 \sqcap A^2_{q,\sigma})\cdots)] \sqsubseteq C_{q_w}$$

for all $q \in Q$, $\sigma \in \Gamma$, and $0 \leq i < n$. Note that this prevents having a leaf labeled with $\overline{W}$ and a leaf to the immediate right labeled with $A^2_{q,\sigma}$. We ensure that the leaves are immediate neighbors by going one step to the right and then only to the left for the first leaf and one step to the left and then only to the right for the second leaf. We also have to forbid the case where we want to do a right move, but are already on the right-most tape cell:

$$L_0 \sqcap \exists S^{n+2}.(G_2 \sqcap A^2_{q_1,\sigma_1} \sqcap A_0 \sqcap \cdots \sqcap A_{n-1}) \sqcap \exists S.(L_0 \sqcap A_{q_2,\sigma_2,R}) \sqsubseteq C_{q_w}$$

for all $q_1, q_2 \in Q$ and $\sigma_1, \sigma_2 \in \Gamma$. Left moves can be dealt with in a similar way. To implement the transition correctly, it remains to state that cells which are not written do not change their content. This is straightforward:

$$L_n \sqcap \exists S^2.(G_1 \sqcap A^1_{\sigma_1}) \sqcap \exists S^2.(G_2 \sqcap A^2_{\sigma_2} \sqcap \overline{W}) \sqsubseteq C_{q_w}$$
$$L_n \sqcap \exists S^2.(G_1 \sqcap A^1_{\sigma_1}) \sqcap \exists S^2.(G_2 \sqcap A^2_{q,\sigma_2} \sqcap \overline{W}) \sqsubseteq C_{q_w}$$

where $q \in Q$ and distinct $\sigma_1, \sigma_2 \in \Gamma$.

We need analogous concept inclusions to verify trees of type 1 and 2, setting concept names tree^1 and tree^2 instead of tree^0, and to interlink these trees in the computation tree. The main difference is that we replace the concept names A^i_a and $A^i_{q,a}$ with $i \in \{1, 2\}$ with concept names that have different values for i, as described above. Details are omitted.

To complete the verification of (accepting) computation trees, it remains to set the concept name A^* from Lemma 47 when we reach the initial configuration. We expect that the root of the initial configuration tree is marked with I and put

$$I \sqcap \mathsf{tree}^j \sqsubseteq A^*$$

for all $j \in \{0, 1, 2\}$. Of course, we also need to make sure that the tree marked by I really represents the initial configuration. In particular, we expect to see the initial state q_0, that the first n tape cells are filled with the input w and that all other tape cells are labeled with the blank symbol. All this is easy to achieve. As an example, assume that the first symbol of w is σ. Then put

$$I \sqcap \exists S^{n+2}.(G_2 \sqcap \overline{A}_0 \sqcap \cdots \sqcap \overline{A}_{n-1} \sqcap A^2_\alpha) \sqsubseteq C_{q_w}$$

for every $\alpha \in \Gamma \cup (Q \times \Gamma)$ that is different from (q_0, σ). To prepare for a simpler formulation of the query, we add the final inclusions

$$G_1 \sqsubseteq G \qquad G_2 \sqsubseteq G$$

which allows us to use G for identifying G_ℓ-nodes, independently of the value of ℓ.

This ends the definition of the TBox $\mathcal{T}_w$. To finish the reduction, it remains to ensure that configurations are properly copied between configuration trees, as initially described. The *i-configuration of a configuration tree* is the configuration represented at the leaves of that tree using the concept names A^i_σ and $A^i_{q,\sigma}$, $i \in \{1, \dots, 6\}$. Note that configuration trees of type 0 have 1- and 2-configurations, trees of type 1 have 3- and 4-configurations, and trees of type 2 have 5- and 6-configurations. We say that two configuration trees are *neighboring* if their roots are connected by the role composition S. We have to ensure the following:

(†) if T and T' are neighboring configuration trees, then the i-configuration of T (if existant) coincides with the j-configuration of T' (if existant), for all $(i, j) \in \{(2, 3), (4, 5), (6, 1)\}$.

For each of the listed pairs (i, j), condition (†) will be ensured with a UCQ, and the final UCQ q_w is the disjunction of these. For simplicity, we concentrate on the case $(i, j) = (2, 3)$. A bit more verbosely, Condition (†) can then be rephrased as follows:

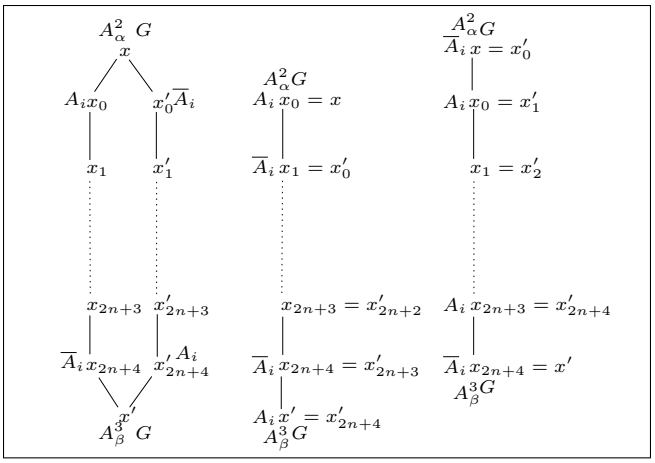

Figure 6: Component query and two identifications.

(‡) if a and b are leaves in neighboring configuration trees of type 0 and type 1, respectively, and a and b are labeled identically regarding the concept names $A_i, \overline{A}_i$, then there are no distinct $\alpha, \beta \in \Gamma \cup (Q \times \Gamma)$ such that a is labeled with A^2_α and b with A^3_β.

We use one CQ q for each choice of α and β such that q has a match precisely if there is the undesired labeling described in (‡). We construct q from component queries $p_0, \dots, p_{n-1}$, which all take the form of the query show on the left-hand side of Figure 6. Note that all edges are S-edges and that the only difference between the component queries is which concept names A_i and $\overline{A}_i$ are used. All variables are quantified variables. We assemble $p_0, \dots, p_{n-1}$ into the desired query q by taking variable disjoint copies of $p_0, \dots, p_{n-1}$ and then identifying (i) the x-variables of all components and (ii) the x'-variables of all components.

To see why q achieves (‡), first note that the variables x and x' must be mapped to leaves of configuration trees because of their G-label. Call these leaves a and a'. Since x is labeled with A^2_α and x' with A^3_β, a and a' must be in different trees. Since they are connected to x in the query, both x_0 and x'_0 must then be mapped either to a or to its predecessor; likewise, x_{2n+4} and x'_{2n+4} must be mapped either to a' or to its predecessor. Because of the labeling of a and a' and the predecessors in the configuration tree with A_i and $\overline{A}_i$, we are actually even more constrained: exactly one of x_0 and x'_0 must be mapped to a, and exactly one of x_{2n+4} and x'_{2n+4} to a'. Since the paths between leaves in different configuration trees in the computation tree have length at least $2n + 5$ and q contains paths from x_0 to x_{2n+4} and from x'_0 to x'_{2n+4} of length $2n + 4$, only the following cases are possible:

- x_0 is mapped to a, x'_0 to the predecessor of a, x'_{2n+4} to a', and x_{2n+4} to the predecessor of a';
- x'_0 is mapped to a, x_0 to the predecessor of a, x_{2n+4} to a', and x'_{2n+4} to the predecessor of a'.

This gives rise to the two variable identifications in each query p_i shown in Figure 6. Note that the first case implies that a and a' are both labeled with A_i while they are both labeled with $\overline{A}_i$

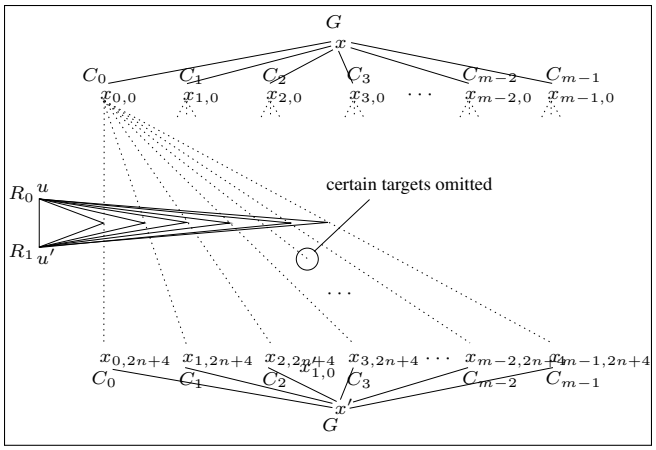

Figure 7: Additional component for CQ.

in the second case. In summary, a and a' must thus agree on all concept names A_i, $\overline{A}_i$. Note that with the identification $x_0 = x$ (resp. $x'_0 = x$), there is a path from x to x' in the query of length $2n+5$. Thus, a and a' are in neighboring configuration trees. Since a_1 must satisfy A^2_σ and a_2 must satisfy A^3_β due to the labeling of x and x', we have achieved (‡).

We now show how to replace the UCQ used in the reduction with a CQ. This requires the following changes:

1. the F-nodes in configuration trees receive additional labels: when a G-node is labeled with A^i_α, then its predecessor F-node is labeled with A^i_β for all $\beta \in (\Gamma \cup (Q \times \Gamma)) \setminus \{\alpha\}$ and with A^j_β for all $j \in \{1, \dots, 6\} \setminus \{i\}$ and all $\beta \in \Gamma \cup (Q \times \Gamma)$;
2. the roots of configuration trees receive an additional label R_0 or R_1, alternating with neighboring trees;
3. the query construction is modified.

Points 1 and 2 are important for the CQ to be constructed to work correctly and can be achieved in a straightforward way by modifying $\mathcal{T}_w$, details are omitted. We thus concentrate on Point 3. The desired CQ q is again constructed from component queries. We use n components as shown in Figure 6, except that the A^2_α and A^3_β-labels are dropped. We further add the component (partially) shown in Figure 7 where again x and x' are the variables shared with the other components, and where we assume that $C_0, \dots, C_{m-1}$ are all concept names of the form A^i_α, $i \in \{1, \dots, 6\}$ and $\alpha \in \Gamma \cup (Q \times \Gamma)$. The dotted edges denote S-paths of length $2n+4$. There is an S-path from every variable $x_{i,0}$ to every variable $x_{j,2n+4}$ except when $x_{i,0}$ is labeled with a concept name $C_i = A^\ell_\sigma$ and $x_{j,2n+4}$ with $C_j = A^k_{\sigma'}$ such that $(\ell, k) \in \{(2,3), (4,5), (6,1)\}$ and $\sigma \neq \sigma'$. The variables u and u' are connected with the middle point of each S-path, that is, with the variable on the path which has distance $n+2$ to the $x_{\ell,0}$ variable where the path starts and also distance $n+2$ to the $x_{k,2n+4}$ variable where it ends.

We have to argue that the CQ q just constructed achieves (‡). As before, x and x' must be mapped to leaves of configuration trees because of their G-label. Call these leaves a and a'. All $x_{i,0}$ must then be mapped to a or its predecessor, and all $x_{i,2n+4}$ must be mapped to a' or its predecessor. In fact, due to the labeling of a and a' and their predecessors in the configuration tree (see Point 1 above), exactly one variable $x_{i,0}$ from $x_{0,0}, \dots, x_{m-1,0}$ is mapped to a while all others are mapped to the predecessor of a; likewise, exactly one variable $x_{j,2n+4}$ from $x_{0,2n+4}, \dots, x_{m-1,2n+4}$ is mapped to a' while all others are mapped to the predecessor of a'. To achieve (‡), we have to argue that $x_{i,0}$ and $x_{j,2n+4}$ are labeled with concept names $C_i = A^\ell_\sigma$ and $C_j = A^k_{\sigma'}$ where $(\ell, k) \in \{(2,3), (4,5), (6,1)\}$ and $\sigma \neq \sigma'$, and that a and a' are in neighboring computation trees.

We start with the former. Assume to the contrary that $x_{i,0}$ and $x_{j,2n+4}$ are not labeled with concept names in the described way. Then they are connected in q by a path of length $2n+4$ whose middle point y is connected to the variables u and u'. In a match to a computation tree, there are four possible targets for u and u' and for the predecessor y_{-1} of y on the connecting path and the successor y_{+1} of y on that path:

1. u, y_{-1} map to the same target, and so do u' and y;
2. u, y map to the same target, and so do u' and y_{+1};
3. u', y_{-1} map to the same target, and so do u and y;
4. u', y map to the same target, and so do u and y_{+1}.

However, options 1 and 3 are impossible because there would have to be a path of length $n+1$ from a node labeled R_0 or R_1 to the leaf a. Similarly, options 2 and 4 are impossible because there would have to be a path of length $n+1$ from a node labeled R_0 or R_1 to the leaf a'. Thus, we have shown that $x_{i,0}$ and $x_{j,2n+4}$ are labeled with concept names as described.

The labeling of $x_{i,0}$ and $x_{j,2n+4}$ with concept names $C_i = A^\ell_\sigma$ and $C_j = A^k_{\sigma'}$ where $(\ell, k) \in \{(2,3), (4,5), (6,1)\}$ together with the labeling scheme of Figure 5 also means that a and a' (to which $x_{i,0}$ and $x_{j,2n+4}$ are mapped) are not in the same configuration tree. Moreover, they cannot be in configurations trees that are further apart than one step because under the assumption that $x = x_{i,0}$ and $x' = x_{j,2n+4}$, there is a path of length $2n+5$ in the query from x to x'. Note that we can identify u with the $2n+2$nd variable on any such path and u' with the $2n+3$rd variable (or vice versa) to admit a match in neighboring configuration trees.

Lemma 50. *$\mathcal{T}_w$, q_w, Σ_w, and A^* satisfy Points 1 to 4 from Lemma 47.*

Proof. (sketch) We have to show the following:

1. If M accepts w, then there is a Σ_w-ABox $\mathcal{A}$ and $a \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{T}_w \models A^*(a)$ and $\mathcal{A}, \mathcal{T}_w \not\models q_w$.
2. If M does not accept w, then for any Σ_w-ABox $\mathcal{A}$, $\mathcal{A}, \mathcal{T}_w \models \exists x\, A^*(x)$ implies $\mathcal{A}, \mathcal{T}_w \models q_w$.
3. q_w is FO-rewritable relative to $\mathcal{T}_w$ and Σ_w.
4. There is an $\mathcal{ELI}$-concept C_{q_w} such that $d \in C^{\mathcal{I}}$ implies $\mathcal{I} \models q_w$.

(1) Take as $\mathcal{A}$ the computation tree of M on w viewed as an ABox, including correct copying of configurations between neighboring configuration trees. Let a be the root of $\mathcal{A}$,

marked with the concept I. The verification of computation trees by $\mathcal{T}_w$ yields $\mathcal{A}, \mathcal{T}_w \models A^*(a)$. Since the copying of configurations is as intended, we have $\mathcal{A}, \mathcal{T}_w \not\models q_w$.

(2) Since the verification of (homomorphic images of) computation trees by $\mathcal{T}_w$ is sound, $\mathcal{A}, \mathcal{T}_w \models \exists x\, A^*(x)$ implies that $\mathcal{A}$ contains a homomorphic image of a computation tree. Note that this tree has the initial configuration of M on w as the root, locally (within configuration trees) respects the transition relation of M, and has only accepting configurations as leaves. Since M does not accept w, the tree must fail to correctly copy configurations between neighboring configuration trees. Consequently, $\mathcal{A}, \mathcal{T}_w \models q_w$.

(3) The query q_w contains only concept and role names that do not occur on the right-hand side of concept inclusions except those of the form $D \sqsubseteq C_{q_w}$. In fact, the FO-rewriting of q_w relative to $\mathcal{T}_w$ and Σ_w is the UCQ $\widehat{q}_w$ that consists of the CQ q_w and (essentially) one CQ q_D for each inclusion $D \sqsubseteq C_{q_w}$, where q_D is the CQ-representation of the formula $\exists x\, D(x)$. This is a slight oversimplification, e.g. due to our use of the markers H and W' used for enforcing that each configuration tree has at most one leaf labeled with a concept name of the form $A^2_{q,\sigma}$. However, it is not hard to see that we can "expand away" these marker concepts, which results in a UCQ to be included in $\widehat{q}_w$. In particular, the markers are propagated only along the boundedly many levels of configuration trees, so the resulting UCQ is finite.

(4) Select distinct $a, b \in \Gamma$ and set

$$
\begin{aligned}
G_1 &= G \sqcap \overline{A}_0 \sqcap \cdots \sqcap \overline{A}_n \sqcap A^2_a \\
G_2 &= G \sqcap \overline{A}_0 \sqcap \cdots \sqcap \overline{A}_n \sqcap A^3_b \\
F_1 &= A_0 \sqcap \cdots \sqcap A_n \sqcap \bigsqcap_{c \in (\Gamma \cup (Q\times\Gamma))\setminus\{a\}} A^2_c \sqcap \\
&\qquad \bigsqcap_{\alpha \in \Gamma \cup (Q\times\Gamma),\ j \in \{1,3,4,5,6\}} A^j_\alpha \\
F_2 &= A_0 \sqcap \cdots \sqcap A_n \sqcap \bigsqcap_{c \in (\Gamma \cup (Q\times\Gamma))\setminus\{b\}} A^3_c \sqcap \\
&\qquad \bigsqcap_{\alpha \in \Gamma \cup (Q\times\Gamma),\ j \in \{1,2,4,5,6\}} A^j_\alpha \\
C_{q_w} &= R_0 \sqcap \exists S^{2n+1}.(F_1 \sqcap \exists S.G_1) \sqcap \\
&\qquad \exists S.(R_1 \sqcap \exists S^{2n+1}.(F_2 \sqcap \exists S.G_2))
\end{aligned}
$$

It can be verified that C_{q_w} has the stated property. □

E.5 Adaptation to Datalog

Our aim is to prove Theorem 17. We first introduce the relevant notions. A *Datalog rule* takes the form

$$R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n) \to R_0(\mathbf{x}_0)$$

where $R_0, \ldots, R_n$ are relation names and $\mathbf{x}_0, \ldots, \mathbf{x}_n$ are tuples of variables such that the length of each $\mathbf{x}_i$ matches the arity of R_i and $\mathbf{x}_0 \subseteq \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_n$. For brevity, we shall speak of relations rather than of relation names. We call $R_0(\mathbf{x}_0)$ the *head* of the rule and $R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n)$ the *body*. A *Datalog program* is a set of Datalog rules with a distinguished relation goal that occurs only in rule heads. A relation is called *extensional* or *EDB* if it occurs only in rule bodies; it is called *intensional* or *IDB* if it occurs in at least one rule head. The *EDB schema* of a program is the set of all EDB relations in it. A Datalog program is *monadic* if all IDB relations with the possivel exception of goal are unary; it is *Boolean* if goal has arity zero. We will concentrate on Boolean monadic Datalog programs. Moreover, we will only use unary and binary EDB relations which correspond to concept and role names from the ABox signature, respectively. IDB relations then correspond to concept names that are not in the ABox signature. For the semantics of Datalog and the definition of boundedness of a Datalog program, we refer to [Abiteboul *et al.*, 1995]. We evaluate Datalog programs over Σ-ABoxes where Σ is the EDB schema of the program. Note that the rule body of a Datalog program is a CQ. Tree-shapedness of a CQ q is defined in the same way as for an ABox in Section 4, that is, q viewed as an undirected graph must be a tree without multi-edges.

For convenience, we repeat the theorem to be proved.

Theorem 17. *For monadic Datalog programs which contain no EDB relations of arity larger than two and no constants, containment*

1. *in a rooted CQ is* coNEXPTIME*-hard;*
2. *in a CQ is 2*EXPTIME*-hard, even when all rule bodies are tree-shaped.*

We start with Point 1, first establishing it for rooted UCQs (a disjunction of rooted CQs) and then strengthening to CQs. Recall the reduction of the exponential torus tiling problem presented in Section E.1. Let P be the tiling problem that is NEXPTIME-complete and c an input for P. We have shown how to construct in polynomial time an $\mathcal{ELI}$ TBox $\mathcal{T}_c$, a rooted CQ $q_c(x)$, and an ABox signature Σ_c such that, for a selected concept name $A^* \notin \Sigma_c$, P has a solution given c iff $(\mathcal{T}_c, \Sigma_c, A^*) \not\subseteq (\mathcal{T}_c, \Sigma_c, q_c)$ over Σ_c-ABoxes.We show how to convert $\mathcal{T}_c$ and q_c into a Boolean monadic Datalog program Π_c and a rooted UCQ p_c, both over EDB schema Σ_c, such that P has a solution given c iff $\Pi_c \not\subseteq p_c$.

It is standard to convert an $\mathcal{ELI}$-concept C into a CQ $q_C(x)$ that is equivalent in the sense that for all interpretations $\mathcal{I}$ and $d \in \Delta^{\mathcal{I}}$, we have $d \in C^{\mathcal{I}}$ iff $\mathcal{I} \models q_C[d]$. We omit the details and only mention as an example that

$$C = \exists r.\exists s.A \sqcap \exists s.B$$

is converted into

$$r(x,y) \wedge s(y,z) \wedge A(z) \wedge s(x,u) \wedge B(u).$$

Thus, a CI of the form $C \sqsubseteq A$ can be viewed as the monadic Datalog rule $C_q(x) \to A(x)$.

The monadic Datalog program Π_c contains the following rules:

1. $A(x) \to \widehat{A}(x)$ for each $A \in \Sigma_c$;
2. for each CI $D \sqsubseteq A$ in $\mathcal{T}_c$ with A a concept name different from A^*: $q_{D'}(x) \to A(x)$;
3. for each CI $D \sqsubseteq A^*$: $q_{D'}(x) \to \mathsf{goal}(x)$.

where D' is obtained from D by replacing each concept name $A \in \Sigma_c$ with $\widehat{A}$. This renaming, as well as the rules in Point 1 above, achieve the separation between EDB and IDB relations required in Datalog. The rooted UCQ p_c is the disjunction of

1. the CQ q_c;
2. the CQ q_D for each $D \sqsubseteq C_{q_c}$ in $\mathcal{T}_c$.

It can be verified that p_c is indeed formulated over EDB schema Σ_c. To show that Π_c and p_c are as desired, it remains to establish the following.

Lemma 51. *A Σ_c-ABox $\mathcal{A}$ and individual name a witness $(\mathcal{T}_c, \Sigma_c, A^*) \not\subseteq (\mathcal{T}_c, \Sigma_c, q_c)$ iff they witness $\Pi_c \not\subseteq p_c$.*

Proof. First, let $\mathcal{A}$ and a be a witness of $(\mathcal{T}_c, \Sigma_c, A^*) \not\subseteq (\mathcal{T}_c, \Sigma_c, q_c)$. Then $\mathcal{A}, \mathcal{T}_c \models A^*[a]$ and $\mathcal{A}, \mathcal{T}_c \not\models q_c[a]$. By the latter,

$(*)$ CIs from $\mathcal{T}_c$ that are of the form $D \sqsubseteq C_{q_c}$ never apply.

Consequently and by definition of Π_c, from $\mathcal{A}, \mathcal{T}_c \models A^*[a]$ we obtain $\mathcal{A} \models \Pi_c[a]$. By $(*)$, the only CQ q from p_c that could satisfy $\mathcal{A} \models q[a]$ is q_c. However, this is not the case since $\mathcal{A}, \mathcal{T}_c \not\models q_c[a]$.

Now let $\mathcal{A}$ and a witness $\Pi_c \not\subseteq p_c$. Then $\mathcal{A} \models \Pi_c[a]$ and $\mathcal{A} \not\models p_c[a]$. From the latter, we get $\mathcal{A} \not\models q_c[a]$ and $\mathcal{A} \not\models D[a]$ whenever $D \sqsubseteq C_{q_c}$ is in $\mathcal{T}_c$. Consequently and since both q_c and all such concepts D contain only symbols that never occur on the right-hand side of a CI in $\mathcal{T}_c$ (except when they are of the form $D \sqsubseteq C_{q_c}$), we must have $\mathcal{A}, \mathcal{T}_c \not\models q_c[a]$. It thus remains to show $\mathcal{A}, \mathcal{T}_c \models A^*[a]$. However, this is immediate from $\mathcal{A} \models \Pi_c[a]$ and the construction of Π_c. □

As the next step, we show how to replace the rooted UCQ p_c with a rooted CQ p'_c. The general idea is to replace disjunction with conjunction. Let the CQs in p_c be $q_1(x), \dots, q_k(x)$ and let $q_i(x_i)$ be $q_i(x)$ with the answer variable x renamed to x_i. Introduce additional role names $g_0, \dots, g_k$ that are included in Σ_c. Then set

$$\begin{aligned} p'_c(x) \quad = \quad & g_0(x, x_0) \wedge g_1(x_0, x_1) \wedge \dots \wedge g_k(x_0, x_k) \wedge \\ & q_1(x_1) \wedge \dots \wedge q_k(x_k). \end{aligned}$$

To make the new query work, we need to install additional gadgets in the toris tree. Recall that every element of the

In particular, we want that for each $i \in \{1, \dots, k\}$, the root of the torus tree has a g_i-predecessor a_i which in turn has, for each $j \in \{1, \dots, i-1, i+1, \dots, k\}$, a g_j-successor that is the root of an ABox which has exactly the shape of q_j. Further, the torus tree gets a new root a_0 that has a g_0-edge to each of the individuals $a_1, \dots, a_k$; note that the torus "tree" is actually no longer a tree. Then a query q_i matches at the root of the original torus tree iff p'_c matches at the new root a_0. The additional parts of the torus "tree" need to be verified in the derivation of goal in Π_c (which is essentially identical to the derivation of L_0ok in $\mathcal{T}_c$). Given that Π_c is a Datalog program and that the rule bodies need not be tree-shaped, it is straightforward to modify Π_c to achieve this.

For Point 2 of Theorem 17, we again start with a UCQ in the first step and improve to a CQ in a second step. The first step is exactly analogous to the construction of Π_c and p_c above. Recall the reduction of the word problem of exponentially space-bounded ATMs in Section E.2. Let M be the ATM whose word problem is 2ExpTime-hard and let w be an input to M. We have shown how to construct in polynomial time an $\mathcal{ELI}$ TBox $\mathcal{T}_w$, a Boolean CQ q_w, and an ABox signature Σ_w such that, for a selected concept name $A^* \notin \Sigma_w$, M accepts w iff $(\mathcal{T}_w, \Sigma_w, \exists x\, A^*(x)) \not\subseteq (\mathcal{T}_w, \Sigma_w, q_w)$ over Σ_w-ABoxes. We can convert $\mathcal{T}_w$ and q_w into a monadic Datalog program Π_w and a UCQ p_w in exactly the same way in which we had constructed Π_c and p_c above. Note in particular that all CIs in $\mathcal{T}_w$ of the form $D \sqsubseteq C_{q_w}$ are such that D contains only symbols from Σ_w, and that also q_w contains only symbols from Σ_w. Thus, Π_w and p_w are both over EDB schema Σ_w, as required. It is straightforward to establish the following lemma.

Lemma 52. *A Σ_w-ABox $\mathcal{A}$ witnesses $(\mathcal{T}_w, \Sigma_w, A^*) \not\subseteq (\mathcal{T}_w, \Sigma_w, q_w)$ iff it witnesses $\Pi_w \not\subseteq p_w$.*

It remains to replace the UCQ p_w with a CQ p'_w. The idea is again similar to the proof of Point 1. However, we now want to avoid introducing rules into Π_w whose bodies are not tree-shaped. This is possible since we work with Boolean queries here.

Apart from the original Boolean CQ q_w, let the CQs in p_w be $q_1(x), \dots, q_k(x)$ and let $q_i(x_i)$ be $q_i(x)$ with the answer variable x renamed to x_i. Moreover, let $q_{k+1}(u)$ be $q_w()$ with u made an answer variable and let $q_{k+2}(u')$ be $q_w()$ with u made an answer variable, see Figure 7 for details. Introduce additional role names $g_1, \dots, g_{k+2}$ that are included in Σ_w. Then set

$$\begin{aligned} p'_w() \quad = \quad & g_1(x_0, x_1) \wedge \dots \wedge g_k(x_0, x_{k+2}) \wedge \\ & q_1(x_1) \wedge \dots \wedge q_{k+2}(x_{k+2}). \end{aligned}$$

To make the new query work, we need to install additional gadgets in the computation tree. In particular, we want that for each $i \in \{1, \dots, k+1\}$, each node of the computation tree has a g_i^--successor which in turn has, for each $j \in \{1, \dots, i-1, i+1, \dots, k+1\}$, a g_j-successor that is the root of a tree-shaped ABox in which q_j has a match. Then a query q_i matches in the computation tree iff p'_w matches in it. The additional parts of the computation tree need to be verified in the derivation of goal in Π_w (which is essentially identical to the derivation of A^* in $\mathcal{T}_w$). This is easy to achieve, but we still have to say what exactly the tree shape ABoxes look like in which $q_1, \dots, q_{k+2}$ have a match. The queries $q_1, \dots, q_k$ are tree-shaped by definition (and use only symbols from Σ_c) and thus we can simply use these queries used as an ABox. For $q_{k+1} = q_w(u)$, we use the concept C_{q_w} viewed as an ABox. And finally, for $q_{k+2} = q_w(u')$, we use the ABox obtained from C_{q_w} by swapping the concept names R_0 and R_1.

References of Appendix

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu Foundations of Databases: The Logical Level Addison-Wesley, 1995.

[Baader *et al.*, 2010] Franz Baader, Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. Query and predicate emptiness in description logics. In *Proc. of KR*, pages 192–202, 2010.

[Chandra *et al.*, 1981] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[Hustadt *et al.*, 2007] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.

[Kazakov, 2009] Yevgeny Kazakov. Consequence-driven reasoning for Horn SHIQ ontologies. In *Proc. of IJCAI*, pages 2040–2045, 2009.

[Lutz, 2007] Carsten Lutz. Inverse roles make conjunctive queries hard. In *Proc. of DL*, volume 250 of *CEUR-WS*, 2007.